Computer Oral History Collection, 1969-1973, 1977

Interviewee: Conference on Data Systems Languages (CODASYL)
Participants: Charlie, Bachman, Dan Fogal, Jack Jones, Dick Kerr, Chuck Greenberger, David Black, Mary Hollis, Greg Dillon, Jim Sweeney, Steve Wright, Bob Bemer, Grace Murray Hopper, Bill Randall, Brian Reynolds, Danny ?, Bill Olle, John Young, Warren Simmons, Bob Grace, Howard Bramberg, Marty Greenville, Stan Epwood, Anaheim Bushed, Tax Metaxides, Bill McGee, John Gosden, Marty Greenfield, Jonas Raven, George Mann, DDF Goodrich Hubert, Ron Hamm, Jim Manner, Herb Beta, Bill Keating, Peg Harper, Dick Schubert, and Herb Manative
Date: May 27-28, 1969
Repository: Archives Center, National Museum of American History

JONES:

The Executive Committee of CODASYL to welcome you here to our tenth anniversary meeting. I need to start out, of course, by apologizing for the delay in getting started, but I guess we're just not very expert at running tenth anniversary meetings.

I'd like to take just a few minutes to introduce the members of the Executive Committee of CODASYL so you know who they are, and make a few comments about the meeting today. The Executive Committee, I'll name them in just the order…I happened to write down their names as they were sitting at breakfast this morning. Charlie Phillips. Charlie, I'm sure, all of you recognize, was the first chairman of the Executive Committee at the time he was with the Defense Department, and was the individual who probably primarily picked up the gauntlet that was flung in on the idea of COBOL and breathed life into it. Bill Ollie of RCA. Warren Simmons of U.S. Steel. Gene Smith of the Navy. Joe Cunningham with the Bureau of the Budget. I guess he's out in the back. Mel Gross of ESO Mathematics. Jim Cooley of Prudential Life. Dick Kerrs of NCR. Bob Curry [?] of the Association of American Railroads. There's Joe Cunningham walking in now. Gordon Bolby [?] of Honeywell. We have a couple of members in America, so to speak—people that served long and faithfully with the Executive Committee, but now have resigned. Greg Dillon from DuPont, and Gene Albertson from U.S. Steel, who is way back in the corner. One member of the Executive Committee could not make it today, and that's Arnie Hestimes [?] from General Motors. He had some other commitment come up, and he had to call to cancel out.

I would also like to recognize three people that have come from ECMA in Europe to represent Europe and participate. As I'm sure most of you are aware, ECMA is our European coordinator or liaison. Mr. John Bessie [?] from ECMA, Madam Shawls [?] from IBM France, and Mr. Bourdain from Bull [?]. Madam Shawls and Mr. Bourdain are members of ECMA's TC6, which is the ECMA COBOL committee.

The CODASYL organization, as I'm sure you're aware, is organized into four committees: the Executive Committee, and then the three working committees—the Planning Committee, which Warren Simmons is chairman; the Systems Committee, of which Bill Ollie is chairman, and who is the one that caused you to be burdened down with the big, yellow book; and then the Programming Language Committee, of which is Dick Kerrs is the chairman, and which does primarily the COBOL type of work. CODASYL is entirely a voluntary and informal organization, and by that we mean there are no membership dues; it's strictly a matter of participation and support by people who have an interest in this type of work and who have been able to convince their employer to sponsor their activities.

The thrust of this meeting is going to be aimed towards what role is there for CODASYL, and I suppose I could say what role, if any, for CODASYL as we proceed now into the next year. I think those of us that do participate actively in CODASYL do it because we do have some interests in some of the things that possibly can be done and contributed to the computer community, but I don't think any of us are sort of free of time and so willing to work that we wish to work on things which are of no interest or of no value or have no place in the computer community. We feel like the need for a common language was recognized appropriately, and COBOL developed and COBOL has enjoyed a significant amount of success. We are concerned as to whether or not there are other areas that need to be recognized and which work needs to be done, and there is, in fact, something that can be done and contributed by CODASYL.

We're going to talk this morning fairly briefly about where we are, and a little bit about what kinds of projects are now underway. We, of course, do have an active program of projects. You will hear about those and have them explained. But we don't expect to spend any time patting ourselves on the back, talking to great extent about COBOL. We're not too interested, frankly, at this particular meeting, in specific changes—why they examined ??? brought the work from the other side. You know, the type of comments that relate to specific changes to COBOL. That's really not our interest at this meeting. What we are really interested in is to have a free and open discussion of what areas should CODASYL should be interested in. The projects that are underway may or may not be of interest, and we'd really like to know. We are concerned that the work that is done and contributed is, in fact, something of interest and something that can be worthwhile.

When you came in, each of you was given a set of two papers paper-clipped together. One of them is the program which outlines the sessions and the people who will be speaking. These questions are aimed only to stimulate the discussion. The people that are speaking are speaking their own opinions. There is no CODASYL position that we are bringing to this meeting. We're not coming with a plan of action and asking you to approve it, or anything like that. We are coming, in fact, with no official CODASYL position. So the people that speak will be speaking in regard to these questions. It will be their own opinions, and we're very interested in your opinions.

The format of the meeting will be essentially there will be several speakers. They will speak for perhaps 10 minutes in relation to the questions that are asked in their session. Then there will be, we hope, ample time for open discussion from the floor. We are very, very interested in your comments. There will be a session monitor. It will be his duty to try to hold comments from the floor to about three minutes, to try and give numerous people a chance to speak at these times. According to the schedule laid out, we do have lunch planned on both days. Coffee breaks. And, as I think is announced there, in the South American room, which is sort of right off the corner here, I believe, there will be a cash-bar cocktail party. We just couldn't squeeze a lot of free booze into $20. I'm sure you recognize $20 will cover the costs of lunches and coffee breaks, and that's about it.

I think without any further ado then, I'll call on Warren Simmons to start the first session of the morning. We're very pleased you're here.

SIMMONS:

Thank you, Jack. The Planning Committee is a relatively new committee, which some of you may never have heard of, so the first thing we want to do is to make you acquainted with the members. The current members are the Association of Independent Software Companies, and they are represented by Steve Wright of Applied Data Research, who is here. I think maybe it'd be a good idea for him to stand up; some of you get acquainted with these people. The Common Organization, which is the association of IBM users of 1130s, 1800s, and 360s. In this case, I think the single 360-type user is a member of common. I don't know whether he's here or not, but Matt Turner of Cameron Iron in Houston was supposed to be here. I know the alternate, Dave Black, is here, so Dave, why don't you stand? Dave has his own firm in Pittsburgh. He represents the 360 portion of the Common Organization. We have the Cooperative Users of Burroughs Equipment, the CUBE Organization, and Daniel Koval [?] of Chrysler Parts Division is their representative. And very tenuously General Electric 600 Users Association. I thought, up until yesterday, Cliff Decker at Hoffman LaRoche was the member, but I understand Ray Connor of GE in Lynn, Mass is supposed to be the man now, and I haven't talked to him. In the GE 400 Users Association is Bruce Reinhold of the Pittsburgh National Bank, and I haven't seen him, so I don't know if he's here. Guide's [?] representative—most of you know Guide, the commercial users of the 360—he's on his way to Florida to a Guide meeting June 1. By way of vacation, I guess, on the way, John Blanchard, Jr. of CNA Insurance in Chicago. I was hoping that someone else here would represent Guide for the session. I do know that several people here are probably active in Guide. I recall that Bob Beemer [?] from GE is usually at the Guide meetings, so maybe he can do the job. [Laughter]

We have implied that an organization, who I haven't heard from yet, the National Association of State Information Systems, the man by the name of Donald Croto [?] of the state of New York is tentatively going to be the representative. The RCA Users

Association, Harry West of Cleveland Twistville. Harry, I know you're here. From SHARE, the scientific-oriented 360 and other IBM computer users, we have been corresponding and conversing with Ben Fadin, who is the chairman of their External Standards Committee and also the chairman of JUG [?]. This is a rather interesting arrangement, and we feel that both of our organizations are going to profit somewhat by this affiliation. He did manage to get someone here from Bell Labs. I've met the man, and I haven't committed his name to memory and I didn't get a chance to write it down. There he is. Stand up and state your name.

KERR:

Dick ???, Bell Labs.

SIMMONS:

Dick, thank you very much. I apologize for my poor memory. The UNIVAC Scientific Exchange was represented by George Mann. George is here, so George, stand up, then I'll explain that they're no longer the UNIVAC Scientific Exchange; they're just now USE. They dropped the need for the words. George is from the UNIVAC Information Systems Division at Phoenix. And then the UNIVAC Users Association, which is other than 1107, 1108 computers, and is represented by Brian Reynolds of Travelers Insurance, and Brian is here. Brian, thank you. VIM, which is CDC 6000 Users Association. VIM, I guess, is a Roman numeral for 6000. It's represented by Bill Randall of Temple University in Philadelphia, and Bill is here. Bill is going to participate in a lot of programs this morning.

I was hoping to hear yesterday from the focus group, which is a CDC 3000 series. Apparently I was out of the room when the phone call came. The reason I was to hear from them is they were going to recommend or not recommend that their users, for a meeting later on this week, to associate themselves with us. I feel sure that they will recommend. In the case of the NCR Users Group, we haven't been able to find a person yet. They're successfully subdivided so much that it's difficult to get one. I'd like to point out that we are trying to coordinate our effort with the X3441 Survey Committee, and Hart [?] Fletcher of that group and myself are exchanging information.

I think there are probably many other organizations that could be effective members of the Planning Committee, and I want to make that open invitation to all of you to nominate an association that you think might become a member. Put that name of the association on the questionnaire we turned in, and I'll make sure to get it and try to invite these people to join us. Incidentally, while I'm at it, if you fail to think of something that you want to say today or tomorrow, or fail to corner somebody and tell them personally, you can write to CODASYL, to the Planning Committee, to the Executive Committee, to the Program Language Committee, to the Systems Committee—any of them—to Box

124 in Monroeville, Pennsylvania. And that's 15146. We'll make sure that your mail is forwarded properly.

Now, about the Planning Committee itself, it has had two meetings, and basically begun to get acquainted. We're gathering information about each of the users associations to find out what their purpose is and their general format and how we can, perhaps, coordinate the CODASYL program in a way that we can keep more users posted. We recently got a survey from Dr. Hopper's group that we jointly released and we think we'll be doing more things of that nature. Copies of that went to the members of the Planning Committee, and with the intent that they would be able to distribute it very widely within users associations. Some of the program this week, I think, we'll take the blame for, as well. Certainly the output that you give us will be very much of interest to our group. We'll have approximately two meetings a year, and usually they'll fall between the various users associations meetings. So we're not going to ask for a lot of travels; it won't be a Programming Language Committee-type commitment from users associations. We'll rely on the mail and telephone.

What is our explicit purpose? Well, we're to gather assimilate and disseminate information, which means that this is supposed to help us design data systems, languages, and do analysis on what's required and so forth. As such, we're going to have to call on many of you in the future to assist in some way, probably very soon. I understand that Dr. Hopper has all kinds of questions for us to use, so we will have a creative work program in just getting the answers for them. So thank you very much.

The next speaker is not me. Dr. T. William Ollie, RCA, Chairman of the Systems Committee.

OLLIE:

Thank you, Warren. Systems Committee Status Report. Where have we been, where are we now, where are we going? The Systems Committee is, I think, rather nicely balanced at a crossroads. We have just completed what appears to be a major piece of work. We worked very hard to finish this report in time for this tenth anniversary meeting. As Jack Jones has already mentioned, I'm the person that's responsible for you having to lug around 375 pages of carefully written and carefully edited prose. I apologize for that, but we felt that it was an appropriate thing to make this report available at this meeting. I will have more to say about the report later.

First of all, a little background to the Systems Committee. It is described in far better words than I can put together in the yellow report, which most of you have. There is, I think, a very well-written paragraph in the introduction which gives the background to the Systems Committee. It was formed from a merger of the original Intermediate-Range Committee and the Long-Range Committee. There was a committee called the Systems Development Committee, which worked on decision tables and on the processing of decision table formatted procedures. This led to the DetabX development. The Language Development Committee worked on nonprocedural languages. That work culminated in the information algebra paper, which was published, I believe, in 1962. Unfortunately, I was trying to find the original charter for the Systems Committee, and I have to confess I was perhaps a little lax here; I was unable to lay hands on it. But I do remember having seen it, and 10 years ago, there was a goal set for the Intermediate-Range Committee and Long-Range Committee, which was something to do with the development of a data systems language of a more advanced level than COBOL.

Now, I have to come to you today, 10 years, and also as somewhat of a Johnny-come-lately, and tell you that I'm afraid that that goal has not been met, as all of you now. But I like to think that we are, perhaps at any rate, a little clearer as to what the goal entails. During the last two years, 1967 up to today, the efforts of the Systems Committee have crystallized into a survey of database management systems. We recognized, about two years ago, that there developing an enormous proliferation of systems, which, for the sake of giving them a handle, I recall generalized database management systems—just a tremendous proliferation of such things. Everybody was in the act. I mean, the manufacturers, the software houses, the customers—almost all around the globe, one might say. Intuitively, this seemed to recognize that there was something inadequate in the tools that were accepted as standard tools.

Now, I spend a lot of time talking about these systems, and I've talked to a lot of people about them. I've realized that this is a very hard subject to get a handle on. The Systems Committee decided that in order to gain insight into what these beasts actually were, we would have to conduct this survey. We started this survey early last year, I would say, in the spring of 1968. The problems that we had in the early days of this work were somehow, as the Committee thinks about them now, "Gee, what were we worrying about?" One of the problems was, first of all, what sort of things does one include? What sort of things do you study here? Does one study, for example, natural language query systems? Does one study report generators, RPGs? Does one study such things as index sequential? Where do you start and where do you stop? Do you study such things as tailored information retrieval systems, of which there are myriads, even more than the so-called generalized systems? Many of the systems that we looked at originally had very much of the flavor of operating systems. It's very easy, in looking at systems of this nature, to get involved in a study of things that really are, essentially, operating systems. To be quite honest, the choice of systems that we finally included in the report was not really finalized until January of this year. All through last year we were looking at different systems and trying to decide whether they merited inclusion, and things like this. Very, very hard decisions, because obviously, as well as the technical--We tried to do the thing, and I think we succeeded. We tried to do the thing on a technical level. We're not interested in pushing any particular system. We're certainly very definitely, from a CODASYL point of view, not interested in anything that smacks of fitting a good housekeeping stamp of approval and go out and buy this one, because that's not the name of the game.

The other access of interest was what sort of features do we look for? What are the typical features of generalized database systems? To be quite honest, this particular part of the work, although it represents but a tenth of the report, it does represent at least half the work. One of the decisions we had to make was whether to make a tabular survey or a narrative survey. By "tabular survey" I mean some glorious matrix with "Yes", "No", and "Maybe" written in the intersections. We decided to make it a narrative survey, which was a challenge, perhaps, to the preciseness of our prose. We hope that this particular approach will prove out. It will prove that one can take nine different systems, as we took, and describe them in some common format using some common terminology.

To go back to the feature list, which, as I already pointed out, did contain an enormous amount of work. I feel very strongly that that is really not complete. It's a good skeleton, but I think that having surveyed nine systems, I think one could go back over the feature list and probably double the number of entries in it. I hesitate to make an estimate as to how many types we retyped that feature list. Something like eight or nine, at least. In fact, the final trimmings were just included at our most recent meeting in mid-April. But the feature is really the cornerstone of the survey. It has been seen by some people outside the committee already, and on the whole, the actions have been quite favorable.

What does the Committee learn from the survey? They have, essentially, 12 people that have been working on this thing, studying these systems, and trying to gain insight into what makes them tick. Well, one thing I think we've learned is that, very definitely, there's still an awful lot more to learn. We don't profess to know everything there is to know about systems of this nature. We think that we have gleaned a lot of knowledge. I feel—and I'm expressing a personal opinion here—that this effort has been a tremendous educational value. But, fine, here we are 15 months later, and what do I say? I say, "Well, there's still a lot more to learn." The current report, as I said, does not compare or evaluate systems. It provides a basis for comparison. Anybody in a customer environment, or even in a supplier environment, wants to develop a comparison. I think we'll find the information in the report of great value. We classed the report as being a vertical analysis. Given the feature list, systems are matched against it. For each feature, there is a narrative description of how a given system handles that feature.

Where do we go next? One point, of course. Since these system descriptions were written by nine different people—I mean, each person on the Committee was responsible for a system description, and about five other people ruthlessly edited it. As one of our most senior members remarked at one meeting, being an author of a system description on the Systems Committee is a very humbling experience. [Laughter] But we tried to get out all the subjective judgments. Anything that smacked of, you know, this system does it this way, and we think that's the great. We tried to get that out.

Where do we go next? Well, there's this goal the Systems Committee has had for 10 years: developing a common data systems language, whatever that may be. At the last meeting that we had, which was about five or six weeks ago, middle of April, we discussed this question. We, of course, have been working hard towards the goal of this tenth anniversary meeting, and we felt we had to come to the meeting saying, "We think we should go this way. What do you think?" rather than saying, "We don't know where to go. What do you think?" because we might, with 200 people here, get 200 answers, and that was sort of slightly confusing. So I'd like to read to you a motion from the minutes of the last meeting of the Systems Committee. I can't really class it as a project which is in progress, because it isn't in progress. But it's where we plan to go to next. I'm now quoting from our minutes. "The CODASYL Systems Committee shall propose at the CODASYL tenth anniversary meeting that the next major effort for the Systems Committee be a feature-by-feature analysis of the feature areas across systems, not necessarily limited to the systems already surveyed, with the ultimate goal of gaining enough insight to develop a common language specification."

Now, that's essentially where we stand. I will say, on the Committee we have discussed where we ought to go next. There's a certain division of opinion as to whether a common database system is A) desirable, or B) achievable, or C) some combination of both. It's not immediately apparent as to what the situation is. I'm now expressing my personal opinion and not the opinion of the Committee and certainly not the opinion of CODASYL. It's a very desirable thing over the next decade—we're looking over the decade now—a very desirable thing that the industry do its development in this area in some coordinated way. For instance, in the way in which COBOL has evolved and has become accepted around the world. We're talking about the interface that a man has with his machine. I think that we're going to be interfacing with machines for the foreseeable future of mankind. If we have as many interfaces as we do men, we're in for a lot of frightful problems. So we need a common interface, and I think we need it on a higher level of COBOL. I am not anti-COBOL; I never have been. I think COBOL is a very, very excellent tool. I think the way in which it evolved is excellent. It's exactly the right way in which data processing development should take place. But here we are, looking at 1969 through 1979. If we have even as many ways of communicating with a machine as people across the globe have of communicating with each other, I think we're in for trouble in 1979. So it's the interface between the man and the machine, but that's not the only part of the problem. There's a problem of data transferability. Can I lift this disk pack on which I wrote data using Operating System X, belonging to Machine Y, and transfer it to Machine P, running on the Operating System Q? Usually you can't, and this is a problem of what we call data description or data definition. We have two sessions scheduled for tomorrow. There's a session on data description languages followed by a session on database management. Both of those are essentially trying to identify the problem and asking the opinion of the assembled dignitaries. Where do you think we should go?

Finally, since my good friend, Warren Simmons, issued a little recruiting plea for his committee, I think I'll take a leaf out of his book and tell you something about the Systems Committee. The Systems Committee currently has 13 members, which is a terrible figure to have. Let me put it like this: We are looking cautiously for new members. We are, to be quite honest, not interested in people that want to tack their names on our membership list to get the material that we produce. The technical report, incidentally, such as this, is a thing that is deliberately prepared with great care for distribution outside CODASYL. We have a lot of internal working papers, which are circulated around the Committee. We are interested in people that are technically competent and would like to join our committee to help us with the work. With 13 people on the Committee now, I think, entering a new phase of our work, we could probably absorb another four or five. What we normally do, we like people to come along as guests for a couple of meetings, and then submit a résumé of their background and join us in our efforts. I can't say that we don't work as hard as the Programming Language Committee, who I believe meet five days every month. We meet about six times a year for two days each meeting, except that just recently, we've been meeting every month for three days and working very hard in between meetings. As one of the other Committee members said, we're sort of almost out of steam. You know, we ought to do some work at home.

So, if any of you do know people or are yourselves people that are interested in our work, I would be very pleased if you approached me on this matter. We do not expect individuals to represent their organizations. We're not a political committee; we're a technical committee. We look for people that are technically competent, and we don't care who they work for. We have people from computer manufacturers, software suppliers, customer environments, and even we have a touch of dignity, we have a university member.

I'd like to thank you very much for your attention and give the meeting back to Jack Jones, if he's there to give it back to. Thank you.

JONES:

We have coffee ready in the entryway out here. I think it's best to break right now, and that will give Dick Kerr's an uninterrupted chance at talking to you after coffee. So let's break for coffee and try to reassemble in about 15 to 20 minutes.

[Break]

KERRS:

To explain Jack's last comment, he was concerned that my talk wouldn't last long enough, to at least 11:45, and I assured him that if I did come up short, that I would follow the lead of the rock and roll group in Miami and just take my clothes off. [Laughter]

MALE:

No! [Laughter]

MALE:

Start talking, Dick! [Laughter]

KERRS:

Really, I appreciate the opportunity to speak to this group this morning. Assembled here are probably the only people in the world who understand how an organization called the Short-Range Committee of CODASYL could possibly stage a report on its first ten years of work. [Laughter] Much less ask for direction for the next ten. I do not intend to recite for you today from the historical section of the CODASYL *COBOL Journal of Development* for 1968. Everyone here has an experienced hand at reading, interpreting, even strategically misinterpreting the COBOL specification. [Laughter] Instead, I'll report to you the progress that CODASYL has made in the organization of the Programming Language Committee and in the specification of COBOL. I shall outline the work presently in progress, our current plans for the future, as well as some of the problems we are experiencing. I originally entitled the talk, "What Hath Jack Jones Wrought?" [Laughter]

The group charged with the specification and maintenance of COBOL has presented a fairly rapidly moving target to the outside world since its original organization. Since 1959, CODASYL has seen fit to reorganize itself several times. With each reorganization came the inevitable changes in the name of its COBOL group. In July 1968, after discarding names like the "Short-Range Committee of CODASYL," "COBOL Maintenance Committee," and the "COBOL Language Subcommittee," the Executive Committee of CODASYL arrested the international name-changing trophy from the Soviet Union Secret Police. By switching to the "Programming Language Committee," the fifth change in nine years, the best the MVD could do was three in the same period. The most recent change, however, is proving to be, for that Programming Language Committee, far more than a name change. It has served to involved the Executive Committee more closely with the other committees by making the chairman of each committee Executive Committee members. I firmly believe that everyone within the organization today is more aware of the overall conference on data systems languages and its goals than ever before.

Membership rules have been altered to encompass more of the industry, and the formerly wordy stringent rules concerning what implemented user proportion was necessary to do something have been reduced to the simple statement that the structure of the Programming Language Committee shall be such that neither those members representing institutions considered to be primarily in the category of implementer, nor those members representing institutions considered to be primarily in the category of user, shall comprise two-thirds or more of the membership of the CODASYL Programming Language Committee. This has served to help us a lot. From the CODASYL constitution, which everyone may not have access to, the membership is based the sponsoring institution's expressed support of CODASYL objectives and upon the availability of a suitable vacancy within the established membership limitation of 25 in the Programming Language Committee.

An institution, they have only one membership. An institution committed to using or implementing a CODASYL programming language may apply to become a member of the CODASYL Programming Language Committee. Such applicant, upon notification by the CODASYL Programming Language Committee chairman, must attend two meetings as an observer prior to being considered for membership, and subsequent to acceptance, must qualify for voting privileges. All participating institutions must maintain a substantial attendance record to gain and retain voting privileges. In this particular rule, we're probably unique in the kind of organization that we are in that it is voluntary and you don't have to come, and seldom can you impose a rule on a group like this. The secretary will maintain attendance records and notify a participant when an absence on the next regular meeting day will result in his attendance falling below 75% of the last 15 meeting days, and thereupon he loses his voting privilege. Voting privileges are automatically reinstated upon achieving attendance requirements. Failure to regain voting privileges within a reasonable time will result in the loss of membership. So it sounds a little tough, but it does seem to work. The attendance rule has had its effect. For example, at our most recent meeting in New Orleans, all of the members of the Programming Language Committee were eligible to vote at the opening of that meeting, which means that everyone had met that rule. If this is a barometer of interest certainly in COBOL today, then interest has never been higher.

Task group effectively has been greatly enhanced in the organization by encouraging the recruitment of experts from outside the CODASYL organization to work in specialized areas, and this has worked out quite well so far as we do not depend on ourselves to generate all the work or to solve our own problems. The Programming Language Committee bylaws outline the task group's best. Task groups may be established by the Committee to accomplish specific jobs. The task group chairman shall determine whether membership in that task group is to be institution or individual in nature. We felt that this would vary with the problem. So depending on the problem, the task group chairman could best decide whether he should just recruit people without regard to their organizational affiliation or recruit from organizations. Membership on a task group is through application to the appropriate task group chairman, who is the final authority on such a decision. Task group members need not necessarily be members of the Programming Language Committee, nor do they automatically become members of the

Committee by virtue of their task group membership. The chairman of each task group is appointed by the chairman of the Programming Language Committee. The task group chairman will see that the minutes of each meeting are kept and distributed to members of the Programming Language Committee. Members of the Programming Language Committee are qualified, of course, to be members of these task groups.

The publication policy adopted within the reorganization has enabled the Programming Language Committee and task groups of the PLC to publish working papers labeled as such for a community response prior to taking action on specific changes. So this is a new thing, and the organization seems to have a lot. The new policy is also responsible for what we feel is an efficient and regular means of publishing the official CODASYL specifications. Now the Programming Language Committee meetings, following the summer break until the end of that year, are devoted to proofreading the updated manual in preparation for publication at the end of that year. With the present system, each member's manual is updated after each meeting so that everyone does have an up-to-date, current specification. Only in any event that changes for any given year are negligible will a new publication not be made available that year. So that if we could theoretically be in a position to publish more often than that, if for some reason or other we change three-quarters of the document. As most of you already know, the journal is now published by the Canadian federal government as well as by our own government printing office. Faced with the possibility of a Canadian COBOL and U.S. COBOL causing an international COBOL specification gap, we have also expressed our willingness to permit the European Computer Manufacturers Association in Europe and the Japanese Standards Organization to publish the result of our work, if they see fit.

Every committee has the problem of dredging up deserving topics for investigation and worthwhile clarifications in the process of developing others. By this I mean in our discussions of a particular point, very often we discover something, very often of great magnitude, that should be done to the specification but can't be done now. If we let it go at that, we'll probably never remember again the next time we get a chance. It's for this reason that the Programming Language Committee, in order to assure that some attention is paid to each of these areas, maintains what it calls a Topic Guideline List. A copy of it is in the questionnaire, and it's sprinkled throughout the document that you have so far as it works into many of the topics that we're having sessions on in the next day or two. It is upon this list that some of the future planning is based. Not all of it, but at least it gives us someplace to start when we're looking to plan something for the future. It's a kind of Programming Language Committee job jar, really, is what it is. At least suitable topics uncovered in the heat of argument over allied points need not be lost in the shuffle this way.

Other changes in the organizational aspects of the Programming Language Committee that I can see are perhaps more correctly termed my own subjective impressions. There is a different talent, I think, being brought to bear on the problem. As might be expected, but need not really be true but it turns out very well, the majority of implemented representatives have indeed implemented COBOL compilers. The users have themselves used COBOL in real situations. One would think with that kind of setup, and sitting at a conference table, facing the man who had a hand in implementing your COBOL compiler, could be, at best, likened to the Paris Peace Talks or Jin Yen Fu [?]. In reality though, cooperation has never been closer. Seldom, if ever, do you sense a particular institutional policy being pushed. There's a lot of "Let's work this out," so it's a real, I think, honest trial. A majority of the members want us to get the work done properly. Also, things like, "Well, how did you implement it?" and to having the advantage of finding out from everyone's mistakes in implementation and use. "How would you use this particular thing in an application program?" and sit down and really hear from people who have used it and who are using it, how they would use this function if it were available. A lot of this talk is traded at each meeting.

The result is, and has been, a healthy injection, I think, of realism in practical experience that can be seen in the latest publication of the specs. Such qualities are not without their drawbacks, as you all know. If anything, we might tend to specify or tend to busy ourselves on technical matters that in fact, in the long run, looking at the big picture, are inconsequential. Seeing the problem against the backdrop of the overall job to be done, we do lose sight of that occasionally. Since the publication of COBOL 60, the specification has been upgraded and republished four times. The first COBOL 61 precipitated by changes rather than new functions. The second 61 extended, introducing the sort and report writer features. COBOL 65, containing the mass storage functions and a different approach to table handling. The major addition contained in the *Journal of Development* for 1968 was an inter-program communication facility. A lot of other changes, but these are just major points.

In August 1968, the U.S.A. COBOL standard became a reality. The novice in the COBOL business was just beginning to understand the difference between CODASYL COBOL; COBOL A, B, E, and F; Stage One/Phase Two COBOL for toddlers; and COBOL for adults only, when we rang in the U.S.A. standard COBOL. [Laughter] We hope that our most recent publication will at least help make the distinction between CODASYL development specification and the U.S.A. standard document. In the introduction, in the paragraph on standardization, we explain, "The effort to define an international COBOL standard was begun in October of 1962 by the ISO Technical Committee 97. Computers and information processing, Working Group E programming languages." I hope the novice doesn't read this. "Working Group E is now Subcommittee Five. The effort to define the United States COBOL standard was begun in September 1962 by ASA, now USASI X344 Task Group on processor specifications in COBOL. This latter effort resulted in a U.S.A. standard COBOL, X3.23, which was approved in August 1968. These standardization efforts as based on the technical content of COBOL as defined by CODASYL." Some of you are getting an objective look, I think, at what this might look like to the outside. But the try is being made. The standard is mentioned in our document.

We're trying our best to keep the specifications separate, and also to enunciate somehow our intentions of how to work together. In no way does the Programming Language Committee of CODASYL, or CODASYL itself for that matter, intend to compete with the various standards organizations. In fact, the cooperation among these groups has been truly gratifying and effective. The Programming Language Committee view of this relationship and how it affects our work is presently being formally drafted for Committee approval. If I may anticipate approval, which I know I may not, but I won't hear about it till next month. I might paraphrase the proposed statement, that the development and maintenance of the COBOL language is the responsibility of the CODASYL Programming Language Committee. Standardization of COBOL in the United States is clearly in the pervue of USASI Committee X3. There is more to the statement, but in fact, that's the point we're trying to get across. The problem of how much attention should we pay to the standard, I believe, is best summed up in a proposed standing rule for the Committee. We're going to be faced with the problem in the Programming Language Committee of a thing that we know should be added to the specifications. But if it becomes part of COBOL and someone implements it, or some user demands it and it is implemented, it will invalidate the standard. How can we operate with this?

We propose that the Programming Language Committee of CODASYL will not make changes to the base COBOL specification that, for no reason, conflict with the standard. However, in fairness to that segment of the industry interested in the continued upgrading of COBOL, and of any other CODASYL-sponsored language, the Programming Language Committee further recognizes that conflict with the standard is not in itself sufficient premise to one, defeat a well-justified proposal. If we really need the thing, we may as well put it in there. We will the well-justified, well-organized proposal to the point of causing already belabored syntax, and explanation thereof, to be more cumbersome than necessary. This could happen quite easily, particularly in the Programming Language Committee. Really, we tried to reword it to keep it from conflicting with the standard and waste three days in the process. Argue at length during the valuable Committee time to establish the exact degree of conflict. Either it conflicts or it doesn't, and we'll trust one another to realize the degree of the conflict. Or give such conflict veto power either explicitly or by precedent. This is another committee psychology. Committees tend to go through fads. We had a witch-hunt that we called where we purged all the witches and chained them to vats. It is really true. I mean, it could catch on to this sort of thing. We could easily get ourselves in a position where somebody, with ground or no, could say, "But that conflicts with the standard," and that would be it. And we don't want to be in that particular situation, either.

Whether or not this relationship with USASI can be improved, and precisely how the status of the various official COBOL documents can be further clarified, are two of the questions I'm sure we'll touch upon this afternoon or tomorrow. Touch upon? We'll probably clobber them. We've been busy since the journal's publication also, clearing away many ambiguities and adding, as a major function of the specification, a communications facility just at our last meeting. We still have a full agenda full not only with proposals generated from within, but with many still coming from ECMA and from the Japanese. The task groups that presently have active include a Report Writer Task Group headed up by Larry Guin [?] from RCA and Jay Sitka [?] from IBM, reworking the report writer specifications. It seems that major surgery was agreed to by everyone who tried to implement it, and they're nearing completion of their work. It was a major job. The Database Task Group, headed by Tax Metaxides from AT&T, who has undertaken a huge task. I know we'll talk about this subject several times in the next two days. The Asynchronous Processing Task Group, headed by Jerry Corpru [?] of UNIVAC. Again, reworking the asynchronous, but used to be called mass storage. It came in with mass storage, the asynchronous processing portions of the COBOL specifications, and another area where the specification appeared to be a lot easier than implementation of them. The Communications Task Group, although they have finished part of their job, is headed by Ron Hamm of Honeywell, is still in existence and wants to continue working in the communications area.

It is a real problem. The two things I mentioned here, the report writer and the asynchronous processing, are interesting in terms of that they are two things that were put in the language. Apparently, everyone thought they would work fine. In the case of asynchronous processing, it looks like it will work, except it was put in at a time when not many people had the ability to actually do this. We're doing it commonplace. It wasn't a commonplace occurrence, the random process or asynchronous process. After it became commonplace, we found that our guess at what common language would be was not too good. Most of the other specification deals with things that have existed for a long time, when you think about it. I don't think we've done too well at guessing about what the future is going to produce, and how we interface with it. Input/Output Editing and Data Representation is another task group that is presently headed by Pete Ingerman of RCA. They've produced several proposals and are continuing to work in this area.

The topic guideline list, as I suggested before, is partially reproduced in the schedules and questionnaires that were handed out. I'll give you a brief description of each one of them. The description is not included in each, and I think it would be interesting. The title doesn't always give it away. The first is debugging tools. Should we? Or will we? Or how? To provide, at the source language level, functions that will facilitate COBOL object program debugging. We've gotten so realistic that we're even thinking about running the object program after its compile. Executive interaction: to provide the ability to communicate with an operating system. That won't tell you much. But in particular, supply the ability to affect a multi-programming environment, perform and reestablish a rerun, communicate with any monitor system, write real-time control systems, which incorporate the manufacturer's operating system, recognize the acts on interrupts and act on interrupts, manage memory object time, and change dynamically the sequence of run units in a job. Now, remember, this is the topic guideline list. We'll get lots of flak if you think that we're proposing that these things must be in there. They're just topics that need to be discussed.

Input/output editing. Data representation. This task will involve the development of extensions to the language, which will provide automatic editing of information. This should include the ability to specify the external medium and external format of data. Specify the internal configuration and format of a data item. Translate the data from the external medium to the internal format, and vice versa. We place selected characters within a string of characters on a straight substitution basis. Validate fields for class and contents. Pack and unpack the data item, and de-edit a data item.

Library maintenance is another one of the topic guideline items. This task involves the development of specifications for the format of a COBOL library, the specifications for a source language that allows for the maintenance of this COBOL library. This activity should also include a review of the existing copy specifications. This one is interest insofar as it's typical of a problem that arises all the time. It is how far out away from the object program and the COBOL source program, out of the realm of compilers, should the Programming Language Committee go. Should it concern itself with when an effect here might be termed a utility? Or should it part of a compiler? Is it part of the language? Is it part of the job? Yes or no.

Extended segmentation, data and procedure. This task involves the development of an extension to the language to provide the capability for the segmentation of data as well as procedures. The segmentation of procedures should include the declarative section procedures as well. This effort should also include a review of the perform ALT and GO specification insofar as they interact with these segmentation rules. This appears to be, gee, something that we would really need, but when considered in the light of what is presently available in this inter-program communication that is now available, it is argued that, in effect, permits you to segment data. Again, the question is does it do everything that we want it to do? Do we want to extend segmentation to, in effect, be a little bit redundant with this inter-program communication?

Mathematical functions. Some of you who have been on a committee will recognize these names because they've been on a topic guideline list for a long time. The definitions change occasionally, though. This task involves the preparations of proposals that will incorporate in the Procedure Division of COBOL the capabilities which now exist in scientific programming languages. For instance, FORTRAN. This task will involve the investigation of all sorts of possibilities. One, perhaps allowing standard FORTAN as a set of language, particularly allowable within the COBOL inter-statement, which would save some specification. Or the specification of trigonometric functions—square root, log, exponentiation, etc., etc.—in COBOL.

Floating point is another topic. To allow within the COBOL language the ability to define a floating point number. Everybody can do it now, but it doesn't exactly work out when you try to go from one machine to another. This should the discussion of representation, size, etc. By "default options", we mean the task involving the development of proposals that we define the assumed option in every case. A selection of several options is allowed, and the programmer has not specified any. This is a good argument just to jump on, default or not.

Free-form COBOL. Task involving the development of proposals that would remove the restrictions on the COBOL format, now specified in the journal. There's an item entitled, "Mass Storage Extensions". It's the task involving the development of extensions to the existing mass storage specification, in order to include the facilities of locked, delete, insert, index sequential file formats, etc. In short, to make random access possible within the standard.

Short-hand COBOL. Abbreviations is another item. The development of a method of including a set of abbreviations for words—keywords, data names, etc.—in the COBOL language, which can be used as substitutes.

Realignment might not be clear by its name. This task involves the development of proposals, which will relocate those features currently in one division which logically belong in another, correcting mistakes. For example, the environmental parts of the FD description that somehow escaped to the Data Division.

A COBOL meta-language. The task involving the development of new meta-language constructs that would permit the allowable syntax to be expressed without the cumbersome general formats and/or lengthy syntax rules.

Generalized subscripting. Involving the development of proposals that will extend the existing subscripting capability, such that there will be no limit to the number of levels, formulas maybe contained within subscripts. Indexing and subscripting perhaps might be replaced in favor of one more comprehensive facility, as described above.

List processing. The database. Again, we'll hear more about it. This task involves the development of specs that will allow data to be ordered in well-defined, interrelated structures—trees, rings, etc. This would include an ability to insert, delete, and update records within these structures. Closed subroutines that would allow a well-defined set of procedures to be executed as a closed subroutine. The problem of collating sequence gets beaten about the head quite frequently, and it managed to get on the list also. The development of specs that would provide a method whereby different collating sequences can be considered by a program.

Global common. Does COBOL need the feature to allow the passing of data from one run unit to the next executed run unit in a job?

Use of punctuation characters. The development of proposals that would remove the requirement that a period must end clauses and statements. The communications facility, as we explained, has all of this topic, although there is a task group active. Until the topic is exhausted, we keep it on the guideline list. But formally, the topic involves the development of COBOL language that will provide the facility to process messages from and to terminals via queues. The COBOL specification should only define behavioral characteristics of the operating system needed to support his processing. That's a keyword there. That's awful hard to do.

Variable-length data items. Somehow, we're still not handling those right. That's not exactly a new thing, but we still have the task, which involves the investigation of the possibility of providing COBOL the ability to define a variable-length data item. This effort should take into account LeCours [?] depending LN picture. That sort of thing.

"Else" in implied conditional statements. We've discovered a new animal in the specifications. In fact, any statement with an LM branch or an invalid key was truly a conditional statement, although it wasn't considered as such until recently. The task now involves the development of proposals that would allow the "else" construction presently allowed with "if" to be used in all conditional statements, including read and write.

Well, that's that. The list of topics that we presently have. I'm sure there are many that you can add, and then we'll get them in the queue. I think our problem here today is to make sure that we spend our time and our manpower in the right place. We know we can list the items that need attention. It's to make sure we give attention to the ones that are most important. The problems that the Programming Language Committee has now I think are fairly commonplace to committees and the industry, and for that reason, probably kind of obvious. The job of keeping a language like COBOL up-to-date and growing with an industry is a full-time job, but with few, if any, working on it full-time. As new areas come to light and the need for further specification is recognized, we can also recognize that we can't begin to cover everything. We have a lot of people who can very deftly describe the problem to us, but not near enough people who can solve it. Therefore, we must be certain that our sense of values is sound in the Programming Language Committee to put the emphasis where it belongs. We find ourselves consistently specifying support software in a backhanded way by expecting certain functional properties to be in the executives, with which the COBOL programmer must interface. This is logical to assume that we'll have to have some kind of interface within the executive system, but you have different executive systems. All of them don't have the same interface points, and how close to get to the executive is always a question. As any committee, we find it difficult at times to keep things in proper perspective, as I mentioned before, perhaps because of the technical background that most of the committee members have now. Sometimes we tend to spend too much time in areas that, although they need work, are less important than others. And so hopefully you'll be able to tell us something about it.

I am going to end up short, but I'm going to renege on my promise [laughter]. In concluding, I would like to remind everyone that…

[CODASYL Disc 2]

KERRS:

Ten years ago, the conference on data systems languages evidently made a correct decision to perceive the COBOL idea. We're still far short of the goal that we laid out for ourselves at that time. Ten years have passed, and you still can't write a tape-print routine that works without some difficulty from machine to machine. There are still many very unromantic, mundane problems to solve in addition to providing facilities to use the new tools just becoming available. I don't mean to say that we should not pay any attention to these, but let's remember that there is still a lot of work, real slave work, that has to be done to make what we already know how to do work. The decision of 10 years ago was evidently realistic. I hope that the ammunition that you give us at this meeting, number one is given to us and not shot at us. But I hope that the ammunition that you give us will lead to an equally realistic decision for the future. I'm convinced that one of the reasons COBOL is successful today is that it was possible at that time. You could get started, and you could do it. I'm not suggesting that the things we mentioned so far are not possible, but we have to keep these things in mind to interface with the world as it is, also, today. We haven't managed yet. Thank you very much. [Applause]

JONES:

I'm glad I didn't make any foolish promises.

MALE:

So am I. [Laughter]

JONES:

I wondered if there were any questions that any of you had that related specifically to any of the discussions of the morning in terms of clarifying any of these particular items or relationships. As I tossed out earlier, the morning primarily was intended to be a status report and a little bit of a look at how we're organized to do business, and what we have underway right at the moment from this point on. That is, starting with the afternoon session, we are expecting to make available and have considerable audience participation. So the fact that we've let you sit here and have chewed on your ear all morning is not an indication of the way we're going to run these two days. Warren?

SIMMONS:

I'd like to take the opportunity to correct one oversight this morning. I asked Irving [inaudible] to stand up. He represents the Honeywell COBOL special interest group, which is the only Honeywell organization that's nationwide. It's sort of fractured, like NCR, also, but they have some clue about COBOL. Thank you.

JONES:

Are there no questions? Yes?

BACHMAN:

I'd like to ask a question. I'm Charles Bachman of General Electric. There are two major efforts going on now which interest me. Two basic questions I'd like to ask, one with respect to the communications work going on with the Programming Language Committee and the Database Task Group. The question that comes this way. I don't think any of us entertained building a COBOL interface for a COBOL communication system. I think we'd like a communication system which could be used to request any program on our machine. So we could have a FORTRAN program talk to the database or a FORTRAN program also communicate with the communication system. I want to know where within the organization of CODASYL it has come to integrate across other languages come about.

JONES:

That is one of the very specific questions we want to try and get answered in the next day and a half, on two ways. First of all, to try to get some feeling for what extent, if any, CODASYL should get itself into this area, and then in what ways, trying to draw some ideas as to, perhaps, approaches to this and so on. I think that's a very pertinent question, and exactly one of the kinds of questions we hope to have discussed and hopefully some sense of answer given to it. I don't think we have in CODASYL, right at this point, an organization specifically with that assignment of working in this area, or even defining it. That's because of areas like this that we're trying to get together on. Do you have a comment, Dick?

KERRS:

I was just reminded that presently, in reality, we're specifically staying away from interface or specifying anything about languages outside of our own. Right or wrong, that's the way it presently stands. We say we can't specify anything about the language that you enter—for instance, in COBOL. And we've said, "Well, we'll live within the environment that we have credited." This is probably wrong, but again, that's the problem. Let's have the solution.

BACHMAN:

I think there are two issues, though. One is the right to retrieve it. The other issue is the media you write on, because the media is not the language. If you put something on the disk file or you put something on a communication line, the format of that, the nature of it, is going to be important to come across the other side. So we need to have specification of what they look like, independent of the language which invokes those questions. So it's one thing to say we'll be cowards [?] and try to stay away from the rest of the world and maybe try to ??? and assume it all. But I think we've got to recognize other languages will exist. Other people want to get on the communication system or get into the same database, and maybe use FORTRAN, ALGOL—whatever languages might exist for that purpose.

FOGAL:

I'm Dan Fogal [?] from Chrysler. I'd like to comment in much the same vein, and it seems to me you can go both directions. You can design a database, or if you want to call it a data-processing database in which everything is nicely composed of fixed-length strings of characters. Maybe 80 of them, for example. And argue about defining that so I can talk about it in an ALGOL or FORTRAN program or get things out of it or write into it, and do my communications. There's also the possibility that some of that database might be binary or it might not be nice character strings, and so the thing cuts both ways. I think a lot of us already have the problem of taking COBOL programs, which are trying to work with data which was not written by a COBOL program. And so you're really going the other way—it's not in both directions at once. You just can't assume, especially if you start interchanging data with other installations, that your database and the language that you are using are the same languages and the same method of operating upon it, that the data was either put there with, or will be used by.

JONES:

I think these are the kinds of things that we will be in a much better position to talk about tomorrow afternoon, after we have gone through the several sessions aimed at a specific discussion of database and of data descriptive languages, and the whole question of procedural versus nonprocedural and so on. I'll hope these same questions will come again, and more like them, after the whole group has had a chance to hear and participate in discussions in more detail on each of these subjects. I don't have anything else to talk about. I'll be glad to try and get questions answered. I don't intend to stand here, though, and stutter and carry on and so on. Just trying to keep here to a quarter to 12. When we're done talking about the morning's work, we'll just adjourn and meet at lunch. Do you have a question, Chuck?

GREENBERGER:

Yes. Chuck Greenberger from Esso Mathematics Systems. On a historical note, I wonder if anyone on the Committee would be prepared to comment about the experience with exchanging COBOL programs across manufacturers' lines. I think this kind of relates to the question that's in my mind as to what among new developments. Is it important to specify, in a standard way, across manufacturers' lines?

JONES:

I'm right in the middle of a considerable experience within a manufacturer's line. [Laughter] We have about 2,500 programs, including quite a number of real-time programs written in COBOL. We've written a program to translate them from the so-called second quarter or the so-called third generation. We've written another program, which takes the old data descriptions into translated data descriptions and generates and a program to translate the data tapes. What we are in the midst of doing right now is capturing production jobs off of the sole second-generation equipment, translating the program, translating the data tapes, running a parallel. Lo and behold, a very large part of the time, you compare the output tapes or the output products, and they're the same. We're making this conversion on about these 2,500 programs, including the real-time system, in about six months with about 25 people, which is considerably better than a conversion we had off the 705 in auto-clear [?], which took that many people about a year and a half. That's the kind of experience we're going through right now. I'm going to have to get very honest shortly because in about three more weeks, we cut onto the new hardware, and we'll see how all these converted programs run in production, even though their parallel is okay. Anybody have across-the-manufacturer-type experience in compatibility problems?

BLACK:

We've taken a real different approach. We're working on Honeywell, IBM, and RCA. We'll be going on a GE system to measure programs. We've taken a subset almost of COBOL for the Procedure Division. We're going with many of the idiosyncrasies it involves ??? ???. We have several programs ??? the Environmental Division. We'll eliminate some of this computational ??? ??? ??? and all this variety scripting terms to solve this. We don't have any real major impeditive besides the problems we had in Israel before we wrote out special programs for the Environmental Division. I've taken the Procedure Division and working almost on a distinct subset of COBOL, but I think we've eliminated many of these problems of getting different answers when we get down at the other end.

JONES:

Would you identify yourself, please?

BLACK:

David Black, DR ??? and Associates in Pittsburgh and also ??? ???.

JONES:

Thank you. Mary Hollis?

HOLLIS:

Mary Hollis, Information Systems ??? Incorporated. We are also taking COBOL across various types of ??? COBOLers and operating systems, and we also use, primarily, a subset. We also find that whatever is poor resulting code on one system, according to the COBOL format, it is likely to be poor on another system. I thought you might be interested in that. We do use a subset, and we attempt to make the most of the conversion within our own systems so that the user can type, more or less, the same thing.

JONES:

I think the Programming Language Committee would be particularly interested in—and I don't want to speak for you, so sit me down if this is not true. I think it'd be particularly interested in those things that cause particular problems in going across machines, and those things that are particularly helpful. But it's the particular problems, of course, that we'd like to iron out.

HOLLIS:

There are some minor things. Really, and totally, as you go, you normally think that things are upward compatible, this thing isn't necessarily so. Very interesting, though. Across the various manufacturers.

JONES:

Greg?

DILLON:

Greg Dillon of DuPont. We have a few programmers admitting to the fact that a particular manufacturer machine gets very fast turnaround due to administrative rules on what can go on. They debug COBOL programs on that manufacturer's machine, and then run them on another manufacturer's machine.

JONES:

Introduction.

DILLON:

Introduction, yeah.

JONES:

Do you judge the performance of the managers, the operations of these two different machines on their turnaround?

DILLON:

I don't quite understand.

JONES:

Oh. It was facetious.

DILLON:

Oh. The turnaround due to administrative will be better run with more than two minutes duration on one of the machines during the day, like ours. That's when they get good turnaround. The other machine now runs up to two hours.

JONES:

Yes, sir?

SWEENEY:

Jim Sweeney, UNIVAC. I think I'd be interested to hear from some of the people who have made comments on taking programs from manufacturer to another as to how they judge the relative merits of the various things they've mentioned. Do we agree that one of our major problems seems to be not restricting the recording on external media to standard data format? Is this a problem in COBOL? Is this something that should have been relegated in the language? That you would not record on external media, implement or define formats? I also am curious as to whether the references to subsets of the Procedure Division are subsets to disallow implementer extensions. Are there certain features in the specifications that we have today which are troublesome? I have to agree with you that the Programming Language Committee would be particularly interested in receiving any comments along those lines.

JONES:

One of the reasons that we wanted to set up a cash bar tonight was to give people a chance to meet each other and talk to each other about questions, some of which that might be of this nature, because I'm sure we're not going to have a chance for everybody to discuss their experiences in particular to the extent that you might like to get into them. But I do hope we can provide the opportunities through these various get-togethers of talking to the various people. Steve?

WRIGHT:

This is a new topic. Steve Wright, Applied Data Research. I used to get the COBOL Information Bulletin, and then it changed to SIGMA, I believe, and now it's the *Journal of COBOL Development*. Somewhere I disappeared from the list. I wondered if you could tell people who publishes the *Journal of COBOL Development*, and how you could get on their list.

JONES:

Well, these things aren't quite the same. The COBOL Information Bulletin was published by the American Standards Institute, but sponsored, however, by BEMI. In other words, the Business Equipment Manufacturers Institute did the printing and paid the bill and maintained the mailing list and mailed out the COBOL Information Bulletin. Some of the information in the bulletin came from the standards part of the COBOL activities, and quite a good portion of it came from the COBOL Committee itself. This was an effort where the COBOL Committee just supplied and made available to the standards people this information on past proposal and this sort of thing. I, frankly, am not current as to the status of the COBOL Information Bulletin. I know there were discussions in terms of producing it under the SIGPLAN Notices or some such means like that, or some sort of newsletter. The *Journal of COBOL Development*, however, is not a journal in the sense that it is a monthly or a semiannual type of publication. It is going to be a specification manual, which will published not oftener than every year, and not less frequently than every two years. That will be available, just like the original COBOL specifications. It'll be published by the government printing office and available for, like, $1.65 a copy or whatever the price turns out to be. It is the document that was also published by the Canadian government. It is not the same kind of thing as the COBOL Information Bulletin. Bob?

BEEMER:

Bob Beemer with GE. Would that be a subset so that one could look at the *Journal of COBOL Development* and have it marked with shading like the IBM manuals so that that subset would be the U.S.A. standard for COBOL and get it for $1.65? [Laughter]

JONES:

I'm not qualified to talk about that problem because we did have a tremendous problem in publishing the standard. One of the reasons that CODASYL does not have a standard mailing list and publication and so on, aside from the fact that it's a tremendous amount of work, is that being a voluntary and informal kind of an organization, we have no financial resources to do something like that. We don't particularly want them, either. We just have no money to pay for the printing and mailing and so on of this kind of a document. It is a problem. I know that the distribution of information on CODASYL and COBOL has left something to be desired. There's no question about that. We're hoping that we can publish this journal, which represents updated specifications, more frequently than we have in order to keep people advised. For example, the new publication will not have the communications package in it, will it?

MALE:

The journal?

JONES:

The journal. The journal is already at the printers and will be available fairly shortly. But just earlier this month—wasn't it the first week in May?—this new communications package was passed as part of the specifications by the Programming Language Committee. Well, that won't appear in the journal until the next time it's published, which is certainly at least a year away. The CIB is not being published anymore, is it?

MALE:

No. It's part of the SIGPLAN Notices.

JONES:

It is part of the SIGPLAN Notices?

MALE:

Jack, it isn't exactly a part of it. All it has to do is to mention the SIGPLAN Notices so it can come out on a separate cover. And there's no change in publication schedule. That is a sporadic publication as the material warrants. So there's no change in the CIBs at all.

JONES:

Okay, except it will be published as part of the SIGPLAN Notices.

MALE:

As CIB Nine, 10, 11…no. Nine was. And 10 or 11 mentioned the SIGPLAN Notices.

JONES: Well, I think we probably need to explore that more thoroughly, because I know there is not a very good channel of information. I think, despite the efforts, we've wound up about a quarter to 12. Close. At this time, I think we'll break.

[CODASYL Disc 3]

MALE:

I'm sure that we enjoyed Howard Bromberg's discussion this noon. I'm not so sure whether we can distinguish between the rogues and the ogres. The one thing that I have not seen in the 10 years of working for the emperor: money on a table, they kept talking about. [Laughter] Although I've been associated off and on with CODASYL, this meeting—and we've discussed it quite at length—is the first meeting that we've ever had a dollar sign put on it, but we just didn't know how to organize it and how to cope with the hotel, how to cope with the lunches without putting—I think you'll all agree—a modest stipulation of $20 for the two days, which allowed about a dollar for unforeseen situations, of which there have been a number already this morning.

The title for this first session is, "The Potential Software Development by CODASYL." Now, with some fear and trepidation, we used the title. As a matter of fact, it was with considerable fear and trepidation that we really outlined the discussion schedule itself. We felt that we probably had to have some kind of an outline to keep coordinated discussion. But this is what we wanted, and this is what we gathered for together, and that is a discussion. We tried to avoid titles and discussions, except of a general nature, in order to get discussions and the leadership principles from you.

The panel today is the commander, Grace Hopper, who, with her birthday suit on, wouldn't put her label on. [Laughter] She appropriately said, "Well, if people don't know me by this time, they won't know me." So, I will give you Grace Hopper in a few moments. Bob Bemer of IBM, Dick Kerrs, and Bill Randall comprise the panel this afternoon. You heard this morning from Dick Kerrs, the Programming Language Committee, the list of projects. I'll save him till late. These projects, we feel, must be explored for development when resources are available, and that should also be a topic of our discussions today and tomorrow. You've heard from Warren Simmons of the Planning Commission, of the Work With the Users Association, with respect to developments being conducted, several projects with the data system language under consideration. You've heard about the X3 effort and a number of a projects in the solo languages. It's quite obviously that these indubitably should be interfaced. They are the de facto standards set by Dartmouth and General Electric called BASIC. There are many new languages that our programmers and our systems analysts are working with for job recovery. Restart in the general man-machine interface.

I think, though, that we might agree that a keynote should be that there is much more compatibility. There is much more of a cooperative user-vendor action. There's a great deal of increased commonality today than there was when we started in 1959 with CODASYL. May I introduce then Dr. Grace Hopper, who will give us 10 or 15 minutes, and the others, who then will give us 10 or 15 minutes. I'll ask her to introduce them. We will then open up the meeting for discussion from the floor.

HOPPER:

Good afternoon. I'm in the peculiar position in having already been retired once, and the prospect of being retired again next June and being retired again in December '71. Such a prospect induces in one a very great feeling of a need of haste. I have characterized it recently to the Navy as, "You get there just as fast as you can, but don't burn out the boilers on the way." I would ask the same thing for CODASYL because there is one thing I need, and need greatly. I hope that some of this will come out in this discussion. We hear of many languages, COBOL among them. A growing need for languages and need for special aspects of languages. But the basic to all of this, and basic to most of the answers, I think, of most of the questions we're asking here, is the need for a data description language, which is independent of the procedure language, and should be independent of all procedure languages. This is the one thing I want, and hope I will get before I get retired too many times. I'd be willing to start off by being able to write some eights instead of nines and refer to octo, or some B's instead of nines and refer to binary. I'd be willing to start in a very simple and elementary fashion within the present COBOL setup, if just it could be extended a little bit. I'd like, maybe, to be able to put on the front of my data some references to what form the data's in and how it's being written down. Maybe that could travel right along with the data.

I think we do need something very badly in the area of data descriptions. I think we not only need to be able to describe the data itself, but also many types of structures of data. And I'd like one language to do this. I'd like to be able to describe lists, trees, rings, and even multidimensional structures of data. I think with the coming multiprocessors, we will need to structure our data in multidimensions. I have some wild ideas on that subject. If you corner me afterwards, I'll tell you some of them. They're not formulated yet, but at least they're ideas. I do think this is one of our greatest needs, is one way of describing data. If I had one way of describing data, I could then build some kind of a compiling thing, with which one unit would handle the data descriptions. Then I can have many scanners leading down to my tables and lists for many dialects. I have to think of small computers as well as large computers, and I know great, big, comprehensive languages and encyclopedias are not going to be stuffed into small computers. Whereas if I had a compiler, which could have many scanners, I could have many languages on small computers. I think one compiler could be built, which would accept many scanners, if I had a data description language, which would supply all of them with the descriptions of the data.

At first, when we started out on COBOL, I think we concerned ourselves far more with the procedures than we did with the data because we hadn't yet started jumping across generations and across manufacturers. Our problem now is that we can pretty well tell the parallel computer what to do. But the question of telling him what he's to do it on is becoming an increasingly critical one. I think this would help a great deal in answering the question of how to jump from one language to another because the major difficulty in jumping from one language to another is how you describe the data, not the procedures. I can write a sequence of procedures that jumps around, but awfully hard to jump that data. And I think it could be done. I think one of our difficulties with our data management systems is the fact that data does take different shapes, and we need one way of describing it. So I think the thing that I would ask for most from CODASYL and from the computer industry, if need be, is something in the direction of a true, independent data description language. And I don't think it'll be interactive either, because it's awfully hard to interact with a data description. That belongs to the procedure part of the program.

I still think we need to be able to describe our data better and to be able to send the description with the data. With ASCII coming and all the other things that are happening, I think this is a major step we need to take. I also realize at the same time that when we get it, it's going to cost us a lot of money to transfer it to a new data description language. But I would remind you that all industries have gone through major changes. The railroads changed from steam to electricity and diesel. The airplane industry went from propellers to jets. Maybe we also have to go through a rather traumatic and expensive transition. You can't tell, when you begin, where it's going to come up when you get there, and sometimes it's a costly step to get there. I do think we need to face these things, and even if they do look like a major move, the sooner we get started, the quicker we get moving on it. It will be less costly and less traumatic. So I would hope that we get a major effort on a data description language, and that CODASYL will extend itself to provide it and to provide it as a support to all of the procedure languages. I would say to CODASYL, "Please hurry up before I get retired for the fifth time." Thank you. [Applause]

MALE:

Bob Bemer.

BEMER:

I don't necessarily mean this as a rebuttal to Howard Bromberg, but once upon a time, there were three bears. I'll stop right there. Actually, as a part of the data systems language development going on outside of CODASYL today, I've been asked to report on what's happening in USASI X3. U.S.A. Standards Committee X3, computers and information processing, operates under a scope of standardization related to systems, computers, equipment, devices, and media for information processing systems. As with most USASI committees, X3, in the area of data systems language standardization, has not engaged in driving standards via its own development activities. For each language standardized or in the process of standardization, the development group has been outside of the USASI organization. A very recent example known to all of us of this separation is the COBOL standard, where the language was developed by CODASYL and the standard by USASI X3. In spite of this nondevelopment history and standardization procedures that suggest the existence of a candidate language, three current X3 projects related to potential language development, or further development, and standardization are currently active in the areas of data descriptive languages, operating system languages, and PL/I. It is these three projects on which I would now like to briefly report.

In November 1968, an X3 ad hoc committee on Data Descriptive Languages held its organization meeting in New York City. The scope of the activity for the DDL committee, as started by its chairman in February 1969, is as follows: First, to examine present and projected activity toward DDLs. Second, to explore the relationships of DDLs to the data exchange system as a whole. This includes such topic as standard codes and formats, transmission conventions, and operating systems' capabilities. Third, to recommend to X3 what work, if any, could a U.S.A. standard for DDL is appropriate at this time. To recommend a structure of tasks suitable for maximum progress towards the desired goal, and to recommend assignment of responsibility for carrying out of the tasks. At the organizational meeting in November, three subcommittees were formed to accomplish the following: First, survey user needs and goals; second, survey current and projected DDL activities; third, compile any technical factors needed in an attempt to structure the problem. All subcommittees have been active since November, and the full committee will hold its second meeting in July. At this point in time, it would be premature to predict the results of the X3 DDL activity, both in the area of development and standardization. However, I note that Mr. Gosden [?], Chairman of the Ad Hoc DDL, is on the program in Session Number Four tomorrow. I'm sure that he'll provide you with further details at that time.

In February 1969, an organizational meeting of USASI X342F was held in Santa Monica, California. The scope of its activity is to investigate the need for, and the desirability of, a standard computer operating system control language. Further, to investigate suitable candidates for standardization: existing and proposed languages, elements of languages, and functions to be implemented in languages. The participants at the first meeting divided also into groups. One group devoted its efforts to surveying the language functions for control and status reporting offered by major operating systems and single-language systems. A list of some 50 functions were identified. The second set out to determine the types of users whose needs an operating system language would serve, and thus delineate the present, and if possible, the potential application there is for a standard. I understand that nine types of users were so identified. The entire committee held its second meeting two weeks ago in Boston and discussed functions provided by the various operating systems in existence today. Assignments were made, and each member of the committee is responsible to identify a common set of principles by which their assigned operating system may be functionally understood and evaluated. The next meeting of X342F will be held in August in Wisconsin.

In January 1969, after about three years of deliberation at lower levels within the X3 structure, X3 authorized the organization of a composite language development group for a cooperative development together with ECMA, TC10, and IFIPS TC2 of the composite language PL/I. The group is charged with producing a language specification for input to X3 for standardization, assuring that the development is in consonance with existing graphed U.S.A. and international standards. The CLDG will report directly to X3 on its activities. To my knowledge (and perhaps Mr. Phillips can clear this up if I'm wrong), a chairman has not yet been appointed, nor has a date been set for the first meeting. However, I understand that there is a number of interested individuals who plan to participate in CLDG. The procedures for the standardization process of USASI Subcommittee X34, as outlined in the December 1968 issue of CACM, describes six major steps in the processing of standards for programming languages. PL/I was the first candidate to be considered for standardization using this process. However, instead of establishing an X34I committee to development a proposed standard, Step Four of the six steps, X3 formed the ad hoc committee reporting directly to X3, to develop the language, as I said before, together with ECMA and IFIPS and produce a specification for input to X3 for standardization.

While USASI is generally considered as a standardization group as opposed to a development group, the standardization process, at times, actually involves development, or at least stimulates development in groups outside USASI. I believe that perhaps one of the purposes of this session today is to raise the question, "Should CODASYL, in its activities in the coming decade, take on some of the development effort needed as a prerequisite to standardization of data systems languages?" An answer to this and other related questions will hopefully be a part of the direction that the CODASYL Executive Committee receives from this anniversary meeting. Thank you. [Applause]

MALE:

Our next panelist is Bill Randall from the faculty of Temple University. He's also chairman of the CDC 6000 User Group on Business Data Processing.

RANDALL:

I'd like to preface my comments by first noting that I'm not a member of the faculty. I'm with the computing activity at Temple. I would also like to make mention of the fact that I didn't see what I'm going to say until about a quarter before 9 this morning, so I stick pretty much with the script, please forgive me. One of the topics that's under consideration is the possibility of coordinating activities sponsored by the various computer user groups. The CODASYL Planning Committee has been investigating this area as a means of achieving, perhaps, community development of data systems languages within the CODASYL framework. Well, an obvious first step is to survey the user groups in terms of finding out what type of activities they are now participating in. The survey, as it stands now, is also complete at about a quarter before 9 this morning. So, the results have not been analyzed in any depth, but at least I think they're probably quite current.

As of today, 10 organizations have responded, to a greater or a lesser extent, representing users of IBM, Control Data, UNIVAC, RCA, Honeywell, Burroughs, and General Electric. The activities, if you can try to read between the lines, tend to group themselves around maintenance and reliability of specific software and hardware products, identification of additional features that the users would like to see, or of new products in areas where their vendor is not currently providing a capability, and a certain amount of user information exchange. There were about 30 or more different types of activities described in a very general fashion. I picked some of them that seemed to be somewhat commonly pursued by a number of organizations. Not too surprisingly, programming languages leads this particular area with COBOL and FORTRAN receiving the most activity by just about all of the organizations. Five of the responding organizations indicated activity in the area of the PL/I. Two indicated they were, of course, actively pursuing projects regarding ALGOL. And quite surprising, only two organizations indicated any sort of activity with assembly languages, and I can't quite believe that. If most of the organizations are like them, they're generally people by systems programmers who are quite concerned with the assembly language. Another major area is the job control language. The problem of programmer-operator-machine communication, interactive graphics, remote computing, system reliability—and here I take this to mean both software and hardware systems reliability. Quite a few are concerned with the restart capability. And also (again, I don't quite believe this), there are only two who indicated they were concerned with a problem of inter-language processor compatibility.

It would indicate there are a lot of areas that are commonly being pursued by the various organizations. I think the problem of coordinating these activities under something such as CODASYL is a very worthwhile project. But I think like many worthwhile projects, it's going to be quite difficult to achieve. I think that a good many of the organizations, at least speaking for myself, have enough trouble trying to coordinate the activities within a particular user group, let alone coordinating with many other user groups. As I previously mentioned within CODASYL, another approach might be a "general users organization", comprised of representatives from the various groups. I personally don't have a feel for which of these might work. I think that it would be a little bit too much to expect that these various and diverse groups would ban together of their own volition. I think if this does come about, it will be through something like CODASYL or USASI. I also think that it'll probably be quite a while in coming, unless we're able to convince the various groups that we can do something for them. I don't think they have this feeling right now. I think that's all I have.

MALE:

Thank you, Bill. I'm sorry to promoted you to… [Applause] You heard this morning from Dick Kerrs. I'll ask him again to talk to you about the activities and the projects.

KERRS:

I don't have the stamina to prepare two talks at the same meeting ever, and for that reason, I would just prefer that you wander with me through the questions here for this session, and two things particularly. One is oversimplification, in general, seems to be a bad thing to do. Over complication is also just as common, but not so often recognized. What I'll do is go through these things and ask myself some questions, or get an answer if I think I have one. If it sounds intuitive, it's because it is. If I were to go through the topics personally—I'm not trying to present a committee opinion; that's what we're here to formulate on some of the problem areas that we've listed here. If I had to rate them somehow from urgent to important to not important, the word "important", at least, they all need to be done. It appears to me they all are important.

To me, the first one, input/output editing, addressing again the differences in data, as Ms. Hopper explained, is certainly important. It probably, to my mind at least, deserves the majority, probably, of our attention in the immediate future. How far it should go: It consistently gets itself tied up with collating sequences whenever we talk about it in committee. That's further on down the line, but it seems to be often addressed in the same breath. We consistently argue about it. Does input/output editing to COBOL or to the programming language mean that the language should interface with every possible data format and every possible collating sequence? My initial feeling is no. It's impossible. It would be nice, but I feel like it's not feasible.

Debugging tools. Again, I think it's a good idea. Certainly, to be able to debug the programs in a standard, uniform, machine-independent manner is nice, but not so important as getting the ability to write a program in that manner very well first. It seems, to me, to be low on the list.

Mathematical functions, of course. We want to be able to do anything that we want to do in the language. To me, here, the question is one of, "How much trouble should we go to?" I personally am of the opinion that we would be better off providing something like inter-FORTRAN, and take advantage of existing rules, specifications, and syntax rather than try to dream up our own again. Maybe I'm wrong. I'm not a mathematician, but it seems as if that's a simple way. It gets the facilities that the programmer needs, and we don't have to spend a couple or 10 man-years on it.

Shorthand COBOL is worthy, but not until longhand COBOL works. Again, it's just my idea. Disagree. I hope you do. I hope that somebody's mad or something.

Collating sequence, I think, we've covered up there. I don't know that by the time we interface with all the existing things, if we can, much less implement that specification after we managed to formulate it, that some of the collating sequence problem, because of standardization and all, might go away.

The default options, personally, I think no. That's simple.

One thing that permeates all of these, to my mind, is do we want to make a programming language, or COBOL, such that if you write a program in COBOL, it will absolutely operate across computer lines? Or do we want to make a COBOL a tool, when used properly, you can do this with it? There is a difference. I'm consistently faced with this. We were talking this morning about how many users had occasion to go from machine to machine. Well, an implementer is always trying to run the other implementer's COBOL program and benchmarks and everything else, so we run into the same problem. It seems to me that many users—not necessarily ones who are all COBOL, have many machines in their shop, but sometimes small machines—chose COBOL because they said, "I'm machine-independent, and I can hang loose." And then they proceed to write in COBOL with abandon, with absolutely no attention paid to what might cause them trouble in the future. They may as well have written in an assembly language or a vendor-supplied compiler of some sort or other. Further, should one all-encompassing procedural language replace a set of widely used languages? I abhor the idea myself. I don't know. Maybe with unanimous opinion or not, I don't think it's reasonable. That eliminates Question 1.3.

How should they be interfaced with one another? It seems reasonable that if I'm not in favor of 1.2, that we should certainly try to get them together, and that having some kind of standard interface, calling interface, between programs seems to be not only the simplest, but the most feasible of the choices, including "other".

How should development activities sponsored by the user group organizations be coordinated with CODASYL? I think we're off to a really fine start there. I hope that the Planning Committee's own plans and ideas are implemented because we really need this kind of feedback and the availability of a large number to give us their impressions of these kinds of problems. Other than that, I have nothing to say. I hope that we can get some reaction from you along any one of these lines, or another one into the general category, if it's possible. Can we have some hands?

REYNOLDS:

Brian Reynolds of Travelers. I think Dr. Hopper probably, coming up first, hit the nail on the head. I'm not sure if it's a language review, but I bet you we think something in the data definition and data direction to be the bonds that link these languages. Being viewed as procedural or nonprocedural is the direction we must pursue. Being the future trend, looks like people will go towards a nonprocedural language, and we definitely will need data definitions or data direction type of linkage to work in interpretative modes.

MALE:

Thank you, Mr. Reynolds. Yes?

MALE:

I'd like to speak as an outsider and ask a question to the panel. A while ago, the March of Dimes was interested in calling all about saving the soft vaccine, and the March of Dimes had to go out and find another disease. CODASYL has always been associated in my mind with COBOL. I would like to know why, now that you've solved the COBOL problem, you don't disband. Seriously speaking though, why shouldn't this be done under the aegis of ACM, or even the IEEE Computer Group, or especially JUG? Something like that. I'd like the panel to address themselves to why the future belongs under CODASYL.

MALE:

Well, I think that's a very good question. I think one of the reasons that we're here today for the two-day session on the tenth anniversary is to find out if our answers or solutions are just seeking problems. Would you like to say something?

HOPPER:

Yes. To answer a little bit to that, one is I don't think COBOL's finished yet. Far from it. It needs a great deal more. The second is, you say, "Why not ACM?" I think the reason there is they don't get anything done. [Laughter] You say, "What about JUG?" and the answer is the same: they don't get anything done. CODASYL does get something done. We are in a hurry. So let's take an outfit that has proved it can get something done. Isn't that a possible answer? If you want to be academic, then I think you go to ACM.

WRIGHT:

I meant to bring this up this morning, and I thought about it. Steve Wright, Applied Data Research. I think that a lot of people happened to imply to him, that possibly think the same as I do before I participated in CODASYL activities, that CODASYL is ??? associated COBOL, but CODASYL is data systems languages, and COBOL is one of the files. I think that it may have been brought out this morning, but I think it should have been made clearer that CODASYL is for further development of data systems languages rather than COBOL itself. I think that's a definite association. In the mind of the public, between CODASYL and COBOL-- and this is one of the difficulties in deciding what

CODASYL should be doing, in that it should sort of divorce itself on being a COBOL organization, which it has become through common usage.

MALE:

I think there's general agreement there, and I think that one of the things that we're here for today if there is something that we should engage ourselves in in the next ten years. Is COBOL far enough along that we can drop it? And if we drop it, what happens?

WRIGHT:

Can I come back again?

MALE:

Yes.

WRIGHT:

That wasn't my point at all.

MALE:

I'm sorry.

WRIGHT:

I think COBOL is a very valid, worthwhile subject to proceed further and to expand, but I think that is in the mind, like is representative of the programmers or the users, there is a definite association between the CODASYL Committee and COBOL. These are practically interchangeable terms. I think that we should stress the point that this is not the COBOL Committee; this is the CODASYL Committee. That it's all data systems languages, not COBOL. COBOL is one of the products, but this is not the COBOL Committee, which is what I think people generally think of when they think of CODASYL.

MALE:

Well, it has taken up a great deal of our time and attention. Danny?

DANNY:

Danny [inaudible]. I'd like to follow up with Steve Wright. The thing that I feel in the case of the X3F24 particularly is that any group that gets together to look at operating systems might be able to arrive at a condition where you have ??? consensus that this is a set of things that are common among all operating systems, without looking to see whether we really are solving the problems we should be solving in the first place. Whether someone who manufactures could go off on a tangent of the "follow me" type of thing. But really, they will be standardizing something that is not what the industry really needs in far as a user-oriented concept concern. I feel that CODASYL should be looking into saying, "What is wrong with what we're doing now so we can right it before it is too late?" This is true with languages and operating systems that we constantly use in that area of the X34 before it comes up as standard.

MALE:

Bill?

OLLIE:

Bill Ollie of RCA. I'd like to try to, if I can, steer this particular session back on the line because I think the Executive Committee wants to go. We're talking here about potential software developments by CODASYL. Not the population explosion among languages, but the population explosion within a given language. I've seen it happen on Burroughs' ALGOL where we've thrown in goody after goody after goody after goody after goody, and it seems to me that this is a very improper subject for development along a common line for a group like this. The user groups are going to con the manufacturers into throwing enough goodies into the individual versions, so I don't think you really have to put too much effort into it on a common basis. It's going to happen anyway.

MALE:

That was the ALGOL recursive.

OLLIE:

Right. But you get into the problem where I guess the *reductio ad absurdum* of it a thing called "track" in which there—except, I guess, for one trivial example—is no such thing as an illegal program. Any contamination of simples you put together does something. This is great if this is what you want in a research environment, and it's pretty tough in a practical shop. Nicholas Worth tried to approach something like this in a language called OILER [?], and he commented later he had abandoned that approach, because I think he said in practice it was extremely difficult to tell correct programs from incorrect programs, which is Chinese for the fact that you can't tell when you've got the damn thing debugged. One of the things we talk about here is source language debugging tools, and I think this is really important. All the languages we have, just about, have been designed in a universe which is limited conceptually to the idea that the program is going to be correct. Therefore, we are talking about a way to say correct things. I think that it is possible to do some design work on constructing your language, constructing your syntax, constructing your notation so as to, perhaps, A) minimize the chance for error, and B) clarify whether it is an error or not an error. You're making the thing a little bit more obvious. And bringing it back to the source language is one of the things that you've got to do.

This business of having to learn how to assembly language in order to be able to find your bugs because the characters who did the compiler just defaulted all their sticky questions into the assembler. It's really painful. I think this is fundamental, and I guess maybe this leads back to the suggestion that Dr. Hopper made in the beginning, which is, maybe what we need to do is to go back to the conceptual structure that you're working on. I don't think the syntax is terribly important. I've never had any problems with syntax or with whether you say move or you use an assignment statement or use a left arrow or an equal sign or any other garbage, and you can certainly translate that stuff. I think that shorthand COBOL is a trivial problem if you have a well-designed compiler because, certainly, what constitutes the token that I'm going to call moot [?] is not a very difficult thing to change the rules for. But I think the question of what kinds of data structure you can work with, what your nouns are and what they mean, what the pragmatics of your nouns are is a really important question and the most important one, and that's the area that a general group like this can do some good overall because, believe me, we in our users groups can beat on our manufacturers enough to get a particular thing implemented if we know what it is we want and we know why we want it, and you've got some idea that the loose ends have been tied up into the work.

MALE:

Yes, Bob?

BEEMER:

Just a comment on the action of users groups. I agree that the user group can get the extra things that they want in their compiler, but I think it's whether the things become common between user groups that the CODASYL PLC has attempted to introduce it as a general thing, which should be done in a common way. If you've got many, many user extensions, if you will, to a given compiler, and everyone has the extension but all in a different syntax, you're losing something. So I think in terms of debugging tools, if every manufacturer has a debugging language in COBOL, there's probably not too much a reason why they shouldn't all be common.

HOPPER:

I think I'd like to bring up another point I'm making. Debugging, and that's the pre-compiler concept. Because the pre-compiler can be used to provide a shorthand to enforce the standard, local, or the overall standard. Actually, it saves debugging time by getting out some tactic bugs before you get into compilation. I think pre-compilers are beginning to appear and beginning to be accepted, and maybe it's worthwhile for CODASYL to look into the area of putting some of the debugging aids and those things into pre-compilers, and possibly setting some definitions in this area. I do think that the pre-compiler concept is being accepted, and it also serves to eliminate those things which are bad practice. I think it's well known that, for instance, one of my favorite topics, the complex "if" statements in every compiler that I know produce poorer object code than just a sequence as simple "if" statements. So some people have pre-compilers that don't let their programmers use complex "if" statements. Partially also, of course, because programmers usually make errors in writing complex "if" statements. But this pre-compiler area is one that maybe CODASYL should look at also. Maybe it's a partial answer in the shorthand, the debugging aids, and those things. Or do logically combine under the concept of pre-compiler.

MALE:

Mary?

HOLLIS:

Ms. Mary Hollis, ISL. I would like to address myself principally to the debugging tools and the source language. Quite frequently, we have a tendency to think of debugging as associated with the first time a program comes on the air. In the early days, when we first thought of compilers, once a payroll was developed, theoretically we would run it for weeks and there would never be any changes. Now that is not the danger of data processing today. The number of changes in any given program are much greater today than they were five years ago, and it's going to be much worse in the next five years than it is today. In other words, we must adapt ourselves to the knowledge that we must introduce changes which, in turn, means debugging. And we certainly want to be able to do this in whatever language the program was written in originally, which we will call source coding. Okay?

Now, Grace mentioned the pre-compiler. You speak of COBOL, but there's another member of the CODASYL Systems Group who did quite a bit of work on decision tables. Decision tables are useful for a better definition of the problem or the procedure. By using decision tables with a pre-compiler, we can do many things.

Furthermore, we are introducing techniques for improving on the existing COBOL program through using the computer itself without the person involved. With decision tables, if it is written, it can be written for replica but still not be the most advantageous format in which to write it. So therefore, again, we can use the system or the computer, the pre-compilers so to speak, to improve on what the programmer put down. We can also develop at a decision table level so that you don't have to wait until you have this whole new system to start debugging; you can debug in pieces. And furthermore, we can debug in an online manner, much in the same way as we are doing in our FORTRAN statements. So there are many improvements, but these are oriented to the user and the better use of what we have. Now, I think with the manufacturers, which you all know quite a number of them, and you know you can't change them very much or their COBOL compilers except over a long period of time, etc., etc., etc. But we can put into these pre-compilers many techniques that will improve the user's position so that he can get his changes incorporated. I think that this is where we need to focus some of our attention on. Very definitely.

MALE:

Steve?

WRIGHT:

I have to expand on the same question that Grace mentioned on pre-compilers. One of the topics there was shorthand COBOL. There are a number of topics along this line. I think a pre-compiler is a macro language for COBOL along the same line. I can visualize a pre-compiler, which will take the program featuring a number of macros, which may be abbreviations that are maybe much more complicated, with new verbs and probably new statements, which produce standard COBOL. A pre-compiler can do this. I think it's important to realize that pre-compilers have this kind of capability, because if it does have this kind of a pre-compiler, then we should address ourselves to standardizing abbreviations for COBOL ??? because we have a macro technique by which the programmer can specify what his abbreviations are. And therefore, we should address ourselves to specifying what standard abbreviations of COBOL verbs, for example, can be specified. There are a number of directions in which we can go through these pre-compilers. I said that a pre-compiler will produce standard COBOL. That's up to the end director whether to go or not, or whether to go directly into COBOL. But I think the existence of pre-compilers should be recognized by this committee or this group, or the pre-compiler technique, and this should be taken into account in determining whether some of these directions should be pursued. For example, shorthand COBOL, I think, should never be standardized because this is one of the things that can be directly accomplished through the pre-compiler technique.

MALE:

For the record, that was Steve Wright, Applied Data. Mr. Young?

YOUNG:

John Young, NCR. Could we generalize this last comment a little ask whether a legitimate period of activity for CODASYL is in more flexible implementation techniques, extensible languages, and directions of this sort?

MALE:

Any response to that?

BEEMER:

That wasn't what Steve said. He didn't say an extension. He says a different mapping of the input symbols.

MALE:

That was Bob Beemer, General Electric.

WRIGHT:

I said a different mapping with symbols should not be the functions of this committee because the technique itself will provide you automatically with this. Why standardize when you can get it as an option of the technique itself? It shouldn't go in the direction of, let's say, standardizing COBOL abbreviations or new COBOL verbs which can be accomplished through a pre-compiler or a macro definition technique.

MALE:

Mary?

HOLLIS:

Mary Hollis. Again, along the same line, in the use of the pre-compile technique, you can actually adjust to changing needs without having to unstandardize standards or without expending a tremendous amount of manpower that is used for the standardization effort. So definitely, we don't want to standardize these parts.

MALE:

I think we've standardized at a great deal of risk of individuality and creativity. Warren?

SIMMONS:

Warren Simmons, U.S. Steel. I didn't intend to talk at all during the session, but the question I want to ask is based on assumptions I must make based upon the people who have talked as a content of what they've had to say. I would assume, from what has been said, no one here is interested in seeing Dick Kerrs' Programming Language Committee continue to develop the communications ??? to do anything about code ??? ???. Has no opinion whatsoever on default options other than the ones he expressed. Understands completely what IO editing is and feels that what they propose to do is fine. End of comment.

MALE:

I think maybe we just haven't got around in communications. Yes?

GRACE:

I'm Bob Grace [?] from RCA, and I never [inaudible]. I think one of the things the community here should consider is what is going to happen in the '70s. I've been looking at it for a while. Seems to me we have three sort of distinct class of people that are maybe on the order of ??? ??? people that are going to be involved in building large systems. That would be on the order of 10^6 programmers involved; it's probably on the order of 10^8 users. That seems like a large number to have discussions and have groups that are so mad about it ???. [Laughter] I don't think it is beyond the scope of this discussion. I think a big part of the latter two classes has to do with tomorrow's discussion, and multiple application of ??? language kind of a thing. I think it's very important to some people who are trying to stamp out some program to be fundamental. Having done that, move out and dump the thing in the user's lap in some incredibly easy way, and nobody has to know anything about computers and certainly not very much of the languages. However, COBOL has been a programming language. Hard to be one which [inaudible] support, for example, of the ???. This is the primary user opportunity. Better people among [inaudible]. What I mean by that is the following kind of thing. Mary Hollis pointed out something that could be very significant. What's really sad is people spend all their time building programs, and they really aren't always conducting shop. Anybody's who's really honest about it will admit it. A lot of work gets done; others don't go through computers. Maintenance controllers and [inaudible] development thing is a big one.

Now, what have we built up to this point in time? These varied languages like COBOL and FORTRAN and ALGOL [inaudible], get the systems where one goes to language and has access to the Library of Congress to get a program going. Really concurring way through all of these open ???. Seems to me the fundamental thing that needs to be done is to build systems, and some of us have built them, where one of the objectives is to make it nice from the program development point of view. This means that the COBOL language is far from complete. I don't believe that it is a matter preprocessing. I believe one has to take a very serious look at what kind of editing and debugging facilities and procedural capabilities so the job can kind of disappear need to be part of that language. My position is I would like to see not too much work done in turning COBOL into PL/I. Or, FORTRAN is taking off, including that into the PL/I. ALGOL 68 already thinks it is PL/I. [Laughter] I would rather we have a number of people who are fairly happy with COBOL, but they aren't happy with the rest of the system. Now, what do we need from that point of view? I'm thinking a great deal of work to be done by this committee under [inaudible], for that matter, on that very problem: what is involved in building a program in a language such as COBOL? My own ideas on this, besides from the editing and debugging facility, are that a language like COBOL or FORTRAN or ALGOL now lead through processing, interactive front end, and for somebody to e able to do that right you need a remote job entropy system that is essentially a very fast compiler. Then you go for an optimizing compiler.

I don't want to take too much time here, but I want to press this point home. Most people really haven't considered this optimization problem very carefully. A lot of people are discounting, and you might get discounted if you will use it there [inaudible]. But I think the major system is programs written in COBOL or some other language. You got to make it very hard ???. Particularly, we could really build programs and terminals and other systems when we get a lot of the requirements out of these background processors, and you can gather to think more about what we need by building something we're really proud of.

Now, I'd like to pick up something that Dick Kerrs said, and I think it's the key thing, the problem we're going to face. He said one of the approaches here is to get the implementers and the users together. That's why we're in all this problem. That's almost impossible to do. We have people that built COBOL entirely. Those people can't use them, because if we use them to do the systems work, we'd be really hammered on the head. Everyone here in this room knows that. After the system is programmed in any well-known language—that includes ???, Burroughs ??? ??? ???—then I think another question then that this group should consider is if there isn't any such language, do you want to worry about turning COBOL into that kind of language? Because I don't think we will solve the problem in the '70s. Now, the user builds large systems programs; the manufacturer will also build large systems programs and do so in the same language [inaudible].

MALE:

Dr. Hopper?

HOPPER:

I think I want to go back to the fact that one of the major reasons for a programming language was so people could use computers, not programmers. Let's not lose sight of that. We do have a certain number of programmers. Systems programmers are largely in the manufacturers, the software houses, and large corporations. There are an awful lot of small corporations and there are a lot of people who want to use computers. In designing or tempting to take COBOL up to the level of a systems language, let's not forget about all the people that have to write programs and the day-to-day work in the small users establishment. This is the one group we tend to forget most often. I would remind all of you that we must think of them, because there are a lot more of them.

MALE:

Howard?

GRACE:

Can I make one more comment?

MALE:

All right.

GRACE:

I believe we've had two different ways of measuring these things. We cannot build information systems effectively ??? ??? ???, and we've been hit on the head pretty hard on this thing. Our users have measured in only one way up to this point, and that is whether or not the thing works. That's a reasonable measurement. If something doesn't work, we wouldn't care how fast it doesn't run. [Laughter] Nevertheless, in the larger companies, the government, for example, are going to steam into the '70s, spending maybe $100,000 a year on this enterprise, on building their large systems and supporting their many users, who are people—I agree with that. Somebody up there is going to say, "Hey, how come you're spending all this money?" Unless we can implement COBOL pretty well, these other things are not going to get done. [Inaudible]. Not in any way am I gunning against the idea that most people shouldn't know anything at all about programming. I really mean most people as far as we have people ??? ??? ??? people of all kinds. I do really believe, however, that there are going to be more solid systems people than we have around now coming out of this school well-trained in large user systems that aren't different in concepts for large manufacturer systems. The same kind of care is going to have to be taken to build those, and we do need a language. I agree with you that COBOL isn't that language, but I'm delighted to hear it said that that's COBOL.

MALE:

Howard, did you have a comment?

BROMBERG:

I just wanted to agree with Ron, [inaudible] agree with him. [Laughter] It seems that there's an element in truth in what everyone says, obviously. I don't mind [inaudible].

[Laughter] The problem that I see is that we have users and we have manufacturers, the user base of people that you talk about. Unfortunately you can't get all the manufacturers to design functionally a type of ??? across the board. By the same token, right this second you cannot get users to write the same kind of program across the board. Now, Ron made the comment that what we have to do is get the implementers and the users together. CODASYL has done just that: it has the implementers and the users together. Unfortunately, they have implementers sit on this side and the users sit on that side, and they're together in the room and together at cocktail hour, but they're not really together in the pure sense of the technical work. Now, perhaps what CODASYL really should pursue is to put the users in business, put the manufacturers out of business, and put language designers out of business in the following way. You recall in the early days there was a two-fold goal that we had in mind for a COBOL definition. One is to provide a language rich enough to allow the specification for a wide range of data-processing problem solution. Well, we've done that, and that's fairly trivial, I think. We have lists upon lists, and everybody knows what is required.

The other thing was to talk about the basis for programming your change, that thing called compatibility. Well, I think what we could have CODASYL do is to get into the implementation business and to design and implement a transferable COBOL compiler. It is obvious, because you have the user, and they know has to be used; you have the manufacturers, and they will tell you what's coming around in the so-called fourth generation so you can take advantage of that. CODASYL now could make, I think, a phenomenal contribution: eliminate the necessary of reinventing the wheel every time a new computer comes on the market, and provide and maintain and withstand, and whatever, this transferable machine, this transferable compiler as an extension of the machine, and have each manufacturer settle only with the responsibility of writing the tail end, a little trivial kind of simulator.

MALE:

What language do you write the compiler in?

BROMBERG:

In Ron's system language. That's what we'll do. [Laughter]

GREENVILLE:

I'm Marty Greenville of Honeywell. I'd like to address this grouping to what I think is a much more important problem, and one that we seem to be avoiding, although it comes up on the periphery from time to time, and it goes back really to the 1.2, the all-encompassing procedural language. This was quickly dismissed, and ??? ??? controversy because it is certainly right in the middle of a great deal of both technical and political consideration. What's really needed is an all-encompassing means of approaching a variety of problems, crossing many languages. Back 10 years ago, when this organization was born, people at that time really believed that there was a difference between data processing and time processing and systems programming, and most of us are still talking as though we still believe it today. When Dr. Hopper spoke about a data descriptive language, I would hope that you were not talking about a COBOL data descriptive language, but rather one that would cross the environment of many different languages. Now, because 10 years ago there was the thought that there really were differences in the types of people that use different types of languages and the approach to these problems, these languages divided themselves into many separate different camps. USASI took FORTRAN, CODASYL to COBOL, ACM and the European organizations took ALGOL, and the universities took some of the list processing languages. As a result, these camps still exist and there is no home for the cross-environmental problem for looking at these considerations and trying to devise a means by which these languages can indeed work with each other. Well, I would like to think of CODASYL as having a broader responsibility than just COBOL. That CODASYL should take itself in the literal term that it actually stands for. If it doesn't take the bit in its own teeth and take the responsibility for handling the cross-environmental problem, at least take on the responsibility of finding a home for these problems. [Applause]

HOPPER:

When I asked for a data description language, I wanted it for all procedural languages. One way of describing data. Then I can jump around to run the procedures with no trouble.

GREENVILLE:

I'd like to see that, but I'd also like to see an organization that can see that that crosses and feeds all our cares ??? ???.

HOPPER:

I was trying to say that I felt that CODASYL has shown its ability to undertake a job and do it, and that I'd like to see them undertake this job and do it. I just hope it doesn't take 10 years. That's all. I think they can do it. But now that they've learned to do it, maybe they can do it faster this time. I think they could if they start out to do it. Of course, everybody thought that we couldn't do COBOL to begin with, but 10 years has shown you could. I hope that maybe five years will show us a data description language. Of course, I'd rather have it in three.

MALE:

Well, there's nothing minimum the ???. Bob Beemer, I had lunch with him, and he said that he agreed with you in 1959; that three months for the short-term program was a little bit tight. Steve?

WRIGHT:

Steve Wright, ADR. Howard brought up the question of broad ??? with computers, and I do ask a question on that. We're all interested in programming languages. The input to these languages is a series of words and a syntactic structure built around these words. I think that's a very important problem. I have a feeling that these four-generation languages should be handling what we accepting as the bases of our languages. That the input to these languages, which a lot of computers aren't handling 30% or 40% of the time, which is an English or English-like language like JCL or COBOL or FORTRAN. These are English words. They are lousy words to handle, but they are not computer words. I feel that part of the problem is that I would like the four generations of computers to allow me to say, "Get me a word from an input stream." If we're going to do this, we have to design our languages in such a way that the machine, whether it's hardware or microprogramming, I don't care what it is, will give me a word from an input stream. I made some studies of the COBOL language as far as how far this can satisfy. It's pretty good. I could design a machine that would give me most things in a COBOL input program, and I could say get me the next word from the language, and it will give it to me. If I did this, maybe I could eliminate compilers. Maybe I could work on the language itself. But I think that we are responsible in specifying in each of the languages to keep this in mind, that maybe in the future, what we're going to be working with, is machines that can actually give us a syntactical unit in our language, and we have to specify in such a way that such syntactical units are available. In COBOL I can give you a dozen difficulties, like the quotation marks with the non-numerical literals, and things like that. Bob, maybe you can expand on it.

HOPPER:

I think one thing you have to do is infiltrate the IEEE computer group, because the engineers don't yet know there are languages. They're thinking in terms of building LSIs that will, for instance, evaluate a sign polynomial. They're not thinking yet in terms of handling languages. I've been to some of their meetings. I think it's time for some of the software people to get into IEEE and try and tell the engineers what it is we really need. They're not hearing from us.

WRIGHT:

But in order to do that, Grace, we're going to have to standardize our languages so that the units, what we call a word—I think COBOL's syntactical unit is a word followed by a parenthesis followed by a period. I would think the computer would give me the next word, then it would give me the next word, then it will give me the next parenthesis and the next period, and it will ignore spaces. All this stuff.

HOPPER:

But we've got to tell the engineers what it is we need. I don't think we are totally communicating between the language people and the design engineers yet.

MALE:

Quite an understatement.

HOPPER:

I was trying to be polite. And that means infiltrate the IEEE group.

WRIGHT:

We're basically trying to stop your massaging, and this is the kind of stuff that the next generation of computers should be doing for us.

HOPPER:

With the LSIs, they certainly should. But I went to the IEEE meeting on LSIs out in California last year, and this is the not the direction they were thinking. It's up to us to tell them, and maybe helpfully CODASYL can help them.

FOGAL:

Dan Fogal from Chrysler. I'd just like to carry this forward. I think the gentleman from Honeywell threw the same idea. I think you can perhaps rephrase Question 1.2 in a way in which I'd say yes rather than no. I care to know, should one all-encompassing procedural language replace the set of widely used languages? I'd like to suggest that I think that one all-encompassing procedural language should underlie a set of widely used languages, perhaps a larger set, and each one not so widely used. The problem that Steve came back with is the one. We run our object programs, if you will, upon a virtual machine, so you can have a hierarchy of virtual machines. As long as I'm talking to one, which provides the primitive functions that I need, I don't give a damn how many layers of software people and hardware people are under there, and I don't care if they have a shrunken programmer in their LSI. [Laughter] The thing is a logical engine, all right? And we don't care if it works through the magic of modern electronics or by monkey's turning the crank, as long as the answer comes out in the time that we're after. I think we're back to the integration thing.

Okay, so I'm back onto the data description thing. If it is general, because we are really talking about defining a set of primitives which are written up and well-defined enough to validly underlie a family of languages which do things. Then I can have a simple enough language, so I get around to the thing that bugs me, which is that the language does everything. I don't know whether I've made an error or not, and so you can define subsets narrow enough to protect yourself or protect the group doing restricted applications, and you still have the underlying set of primitives which are similar, so that when you make the thing richer, you're not in a different ballgame.

HOPPER:

I just think that in a separate editing of different languages, you want to do that. It must be independent of the procedure. You've got to pull these completely apart. Describe the things in one box and what's to be done in another box.

MALE:

Bob Beemer?

BEEMER:

I'd like to expand on Marty's point over here, going back to something I first rejected, is that we should do it—we should do it, and in what environment. You made a very well-taken point, I think, which could be interpreted as saying we have gone in the direction of these several languages for apparently different functions, and they're not really different cultures. But the question to pose here is not what language exists contra-distinct of what some other language should be, but one of the organizations should get this. As far as I'm concerned, CODASYL, IFIPS, USASI—none of them are limited in their scope as far as the language is concerned. The problem is that your hierarchies and your aristocracies have to build up among the various languages. The problem is how we do we marry the dukes of COBOL with the duchesses of ALGOL? We've got to break down somehow these [inaudible]. I know this group of people who meet in New Orleans every third Thursday, these guys, well, we always meet at SHARE or something like this. That seems to me to be the problem. We have intrinsic self-interest in all these things, and we carry it along for long periods of time, then it gets buried. I've seen some of this, and I don't think anybody that works in these groups would deny it. The problem is not who is going to get the assignment for this thing; the problem is to get the people in the framework to cooperate to blend these languages and somehow get them together.

MALE:

[Inaudible] again. I think that perhaps CODASYL ought then to address them to defining a core language instead of primitives. They ought to be in the basis. At the ACM SIGPLAN Extensible Languages Symposium it was pointed out that if you ever expect to do something, for example, like real-time processing in a higher-level language, you better have the ability to accept asynchronous interrupts in the core, in the basis. A whole set of these primitives, which may not ever be implemented in any two of the languages, had better be at the bottom, at the core. If you get these things defined by one group, it had better have a large share of users from a large share of universities, because one man's primitives are another man's poison.

BOB:

Bob ??? from ???. I'd like to make a point. Ten years ago, this group started. We had basic-level languages, and procedural languages were, as Howard said at a luncheon, people didn't know much about them. Basically didn't even know that you could do it for sure. Okay, we're 10 years further along the line, and there are procedural languages all over the place, each of them designed to do, perhaps, a particular range of applications. And it may be that it's really not possible to take one procedural language and have it applied along all these different specifications. Certainly, Big Brother had enough trouble with that one. But there is another set of things that is in its infancy and might be of a proper area for this kind of group to look at, and those are the nonprocedural languages. Those are the things that are now in their infancy. My personal feeling is that those are the things that we're going to see in their adulthood, or at least in their middle adolescence in the next 10 years, at the end of another phase. That really is the proper area for a group like this to look at.

MALE:

Terry?

TERRY:

I would like to address myself to a minor matter concerning all-encompassing procedural languages, and that is the training problem. It may become impossible to train anybody to use an all-encompassing language. It obviously has to consist of a set of languages. It cannot be a single language, and consequently, means that the [inaudible]. They should be interfaced, of course.

MALE:

Can I comment on that? Over the last two points, there's something to be said for each of them. Obviously, we've got to make it easier for the people who proctor these ??? in order to add to the large business of building very expensive systems. One of the reasons that we kind of wound up on the moon recently is somebody knows something about physics and the language goes with it. I don't believe anybody has ever used mathematics until it was made into something that's easy to learn. It's a language you work a long time to learn. You do that because the problems are hard and you couldn't solve them without the language. The notion of a language, if any more of us can build these large systems, well, you don't try to make it unnecessarily hard. Nothing would lead me to believe that this is something that somebody's going to have to be able to learn in two or three weeks. The problems are huge. We're spending millions of dollars on these things. The language is associated with proper expression and solution to this problem is not going to be ???. It would be helpful if there were a language, so that all of us who know our systems could come to understand what's wrong. To beat the point, it's a real problem. We build COBOL compilers, and I think we build reasonable ones. From an operational point of view, if you don't use the language, you really do have difficulty understanding just what is wrong. Some of these problems could be overcome with a common language. Look at anybody's Assembly. People build software in Assembly. Those assemblers are the ???. Everybody's higher-level language compilers are quite varied and many are invalid. That's because they aren't being used by the people who built them. I think there's a lot to be saved in the maintenance ??? ??? ???. The only thing I'm suggesting here within this framework, and ???'s group to consider that problem, or since there is no such language, where is this going to come from?

HOPPER:

I think we brought up the point of the nonprocedural languages, and I think we're tending to say "a language", and we're tending of languages all on the same level. Shouldn't we be considering hierarchies of languages for hierarchies of people? And don't we need the nonprocedural as well as the procedural? Don't we need the systems level language and the people level languages? Aren't they hierarchies? Not necessarily subset, but hierarchies? Increasingly symbolic and mathematical? Decreasing to the English words' common usage? Let's not lose track of the possibility of thinking in terms of hierarchies rather than-- I keep getting a feeling we're talking about languages all on the same level, and I don't think they belong that way. I'm quite sure what I'm trying to say. [Laughter]

MALE:

And that's the kind of thing ???. I really thought they'd help us with the [inaudible]. Right now, 1.2 talks about an encompassing procedural language through Basic, and why we use languages. Well, [inaudible] is what he chose. The languages in COBOL shouldn't have to be. I think we should then turn around and say everybody will use COBOL before we try the schedule thing. The question broke to a procedural/nonprocedural, the hierarchy of languages. The user should be the one to decide what the [inaudible]. They've got massive volumes to process, massive volumes of data, perhaps we cannot stop the nonprocedural approach, because the less you know, the costlier it is. If you don't have a map to get around the town, then you're not going to find your way there very quickly. So I feel that they should not be ruled out that there would be a parallel development of languages where some of the languages are nearly competitive and others are performed in [inaudible]. But regardless, I think we [inaudible] is the one that I've heard mentioned three or four times. The only [inaudible] is that there should be a data description language. It would be very nice if that data description language was independent of the other languages to interface with that data description. With the data be sent on the ??? ??? of that data description language. So what I think we need is a commonality of data description languages, but not commonality at the procedural level, so that we could interface and have FORTRAN, ALGOL, COBOL all working with the same database. Indeed, I think the procedural languages should also work with the same database, and that is absolutely essential.

MALE:

We have a specialized session tomorrow morning on the procedural versus nonprocedural. We have one session for data description languages. I think that we probably will want to sharpen our thoughts and develop that further. Ready?

MALE:

I want to go back to my memory of years ago this really big roundtable where, I think in the second day, everybody was asked to express their opinion about something or other. I remember Gene Albertson saying, "Well, he didn't care what the language would write, as long as it helps the documentation problem." At that point, I was with him, going to ??? committee. And I wonder if in the area of just taking the COBOL language itself, and I remember a remark made on the floor that any language in the writing of a program was only about 10% of the cost of the job. That's a little bit outmoded now because I don't think that quite much systems work being done today as there was in those days. But there's still around 50% of the costs involved in debugging and documentation, let's say, of those jobs. I'm wondering if the COBOL language people can look more at their using these costs by the addition that they can add to the language to assist in both the documentation and in debugging. Because it is still a very large cost. I believe the language and additional processing of the ??? program can achieve some cost reductions just in that area alone.

MALE:

??? ???. Bob?

BEEMER:

Just a brief remark on that. I checked with IBM some years ago since the 360, and the split between the programming and documentation is, as you say, half and half. But that's really a third and a third, and the last third is the maintenance of the program or the debugging the software.

HOLLIS:

But I don't think enough automation is done if ??? the documentation is [inaudible].

BEEMER:

I said that's what happened to System 360. I didn't defend it. [Laughter]

MALE:

Charlie, will you identify yourself?

BACHMAN:

Charles Bachman, General Electric. Looking at the program of the next two days, there's quite a bit of concern about data description capability and database management capability. I think one of the ??? In that paper about communication deserves a good bit more attention than it's apparently going to be given in the next few days. I think this is a particularly crucial thing that I can look now and see disk file and equivalent kinds of things in great quantities supporting our business needs. I think we're going to come in about as fast as you let communications, even though it's against two of those disk files. The place we started out getting cards, some of us might paper tape into the computer with some trivial computation of the backup of the local device is very quickly going to disappear. The people are not in the computer room. The people are in the same town, even in the same building, in the same United States, but the systems are spread out geographically, and we need considerably more capability to be able to quickly address the result of the program, an output file, if you like. Some place, and more likely, some person that we want to go to.

There has recently been adopted by the Programming Language Committee a report that communicates in task groups. I think this report is very important in terms of its intent to open up the door between getting something from the terminal to a program and getting it back out. I think if we had a chance to discuss this a tremendous amount as additional work be done into generalizing a whole approach, because the source of an input message, the old one destination is something which cannot be determined at the time the programmer compiles the program. It won't necessarily come with tape file or a card file; it will come from all over the United States. Where it goes back will largely be dependant on the result of the computation. So one of the variables in the computation will be who is this source or the final destination of a message, which may be another programmer or maybe it's somebody across the country. I think there's attacked very strongly, because our machine isn't what isolated the world. We had quite a nice system manually 20 years ago that we very quickly just [inaudible] magnetic tapes and lost it. But many people think their businesses are poorer than they were 20 years ago. They feel they can't go back to the volume of data. But we do not have easy man-to-machine relationships. We don't get ??? ??? readily. And I think that this area, if you look at the so-called newer lines like PL/I does not attack this part at all. We're looking for fancier mathematical techniques, fancier ways to control procedures. But not how they get back and forth between people and machines. I think this needs to be attacked, and really is deserving of the whole session this week.

MALE:

Bob?

BEEMER:

I'd like to augment Charlie's statement in this. I looked through the data communication language that was adopted. It's primarily directed. He says between terminals and computers and back to terminals. This leads to a very interesting thing in mass storage. It's been ascertained there are two classes of people that have mass storage needs: those that come up to approximately 16 billion characters, and those that start at 25 billion and go up to infinity. Now, this gap between 16 billion and 25 billion characters, crazy enough, comes with the fact that if you take all the people in the United States and start keying in input via paper cards or paper tape or something, that's about as much data as you can get in here. So that's the reactive base between computer and manual input. But then we come to a more integrated society where one man's output is another man's input, and in fact, computers will generate massive files, which they will then operate upon and they don't have to be produced by human keystrokes. Then the important thing will be how are you going to move these large masses of data back and forth? And it's to this point that the data communication language is, so far, somewhat incomplete.

HOPPER:

I think what the last two speakers have been saying is that throughout all that we do, we have to keep in mind in any given situation the total environment, which involves not just the programming language, but involves documentation, and it involves the people that are using it and the managers of those people. I think we have to realize that those range from the very small to the very large, and that we may even have, in the same environment, small and large. We must keep in mind all of these things when we talk about these languages—the total environment. A language which could be suitable in one corporation would not be suitable in a small business and a small computer. I think we must always address ourselves to the environment in which something is to be used, and that above all, while it may not always be theoretically perfect, we can at least make it useful.

MALE:

Chuck?

GREENBERGER:

Chuck Greenberger, Esso Mathematics and Systems. I have heard a lot of sentiment towards Dr. Hopper today in here about the management facility. I would like to just point out that I think there's a conflict between his position and some of the other ones about procedural languages in multi-facilities and the current Programming Language Committee activities menu, which will be oriented towards extending COBOL to provide extended data management facilities, and more problem oriented language facilities within COBOL. I personally am against these extensions. It looks as if they would proceed, with the support of Dr. Hooper…

[CODASYL Disc 5]

FEMALE:

Well, it worked on COBOL. While they were interested in a computer industry, they worked on it, and they were able to put their time on it because their companies needed that product, and it was commercially worthwhile for them to do so at that time. I think that we have to face up to the problem that as far as committee action, where it is of a voluntary nature, that we do so, providing that it is a worthwhile project and a worthwhile product that is needed within the foreseeable future. You also have to realize that while we are doing this at this particular time, it will also be contributory for the next problem, which is coming up. But let's face it: none of us do work just because of the general interest because it is commercially and dollars-and-cents worthwhile, and because it was needed.

Now, I also think that as of this moment, many of our students who are coming out of colleges are ready for systems that we don't have the tools to get them to. We don't have the tools for them to use at this moment. They are thinking in terms of management information systems, database systems, which are the requirements of business and industry because of the environment that you operate, and I think CODASYL and all the people here must face up to what the needs are because of the business environments. Dollars and cents involved. This is the basis on which we do work.

MODERATOR:

I think that's pragmatic.

MALE:

I think that's the whole [inaudible]. We've got to get down to what they're going to be using down there at the basic levels. I want to try to discourage [inaudible].

HOPPER:

One interesting thing is I've heard some people over here and some others say that they think computers could now be built that would assist in implementing the languages. Ten years ago, we didn't yet have the languages and couldn't ask the computers to be built that way. I'm not totally sure we can predict what will be around 10 or 20 years from now. I'm not sure when they first built propeller airplanes that they thought there were going to be jet airplanes. I'm not sure we can totally know what the future computers will be like, but that we may contribute to what they'll be like. I think we have to start on the basis of what we know and is needed, and move as best we can towards independence. And that's pre-manufacturer, designer, software expert. As we move to a higher level, we free them to move up and give us what we need when we move toward the next level. I hope it's not a static thing or that it stops anywhere or that we ever forget how to change.

EPWOOD:

Stan Epwood [?]. Now that I have declared myself a fragment of this, and come out with philosophy, I feel fairly safe in making the next comments. It just occurred to me that I have heard people talk about preprocessors or pre-compilers, about nonprocedural languages, about what I can only call programming aids or tips on how to use. In considering these comments, I feel about them more or less as enlightened self-interest, which I think has every place here, by the way. I feel that that's one of the most important functions of these ???. Things are being done in the computer business on a pragmatic level, on a in-the-one-machine-shop level. As usual, procedural improvements drift downward in the computer business, from the big companies—companies who support, can afford to support, who must support efforts like CODASYL and USASI and others' voluntary efforts, who can afford to send people who can afford, first of all, to hire people with sufficient experience and expertise. Who, secondly, can afford to send them; who find it necessary to do so for the very pragmatic reasons that you mentioned.

Now, things drift downward in the computer business, and we are at a stage today where a number of computers are making their living. And a number of other people in organizations are contributing equally in these kinds of areas, what we might call peripheral software. I don't know exactly what to call that. It's software ??? in my manual. I think that this is happening because there is a need. We need to be at least partially, if not terribly effectively, fulfill these kinds of thinking. I think that this kind of coordinated input into CODASYL and some of the other organizations is quite in order at this time, because if we're talking about it—I'm not saying that I'm in favor of it—but if we're talking about increased standardization, not only in the USASI sense, but in the sense of a central set of specifications for a language or a set of languages or what have you, then we need a funnel where things can enter and be screened, and perhaps these things that are peripheral today will be more central tomorrow.

MALE:

As the afternoon goes on, I didn't want to see Warren get frustrated. Nobody's made any comment on the fall options and very little comment on source language decoding/ debugging. Well, the next version in a language, these things are conveniences you can do without. To the casual user, they're a necessity. I'd like to make a point that the normal user of a language is a casual user. You introduce the basic language, you get three or four manuals on a language that is simple. You get a little folder, and with half an hour's instruction you can learn to use it. Some features you cannot use. I was making an evaluation recently, and I asked a friendly IBM representative if he had a manual on him, and he comes in with three or four manuals. There are very few manuals on any language that are any way near as small as these documents that were passed out at this morning's administration. To be an expert, you have to understand them. But somebody does get to the point where they understand this. They become tagged as an expert, and I think they move on to this next hierarchy of languages that Dr. Hopper talked about. So I think the person that's constantly upgrading himself, and they are the casual user—their interest is not being an expert on a language, but using it to do a job. You're in that position without the default option, without being able to debug the source language, then you can't do anything; the whole thing's a waste of time. I think no new language, or no useful language is going to come about on the assumption that people on experts when they start using it. I think these things are fundamental.

MODERATOR:

Ron, brief comment?

RON:

I'd just like to comment on source language debugging, which really references a paper that I'm very much impressed with that was given at the Spring Joint Computer Conference written by Bob Walter at Rand on an X band system. So I recommend all of you read it if you have the opportunity. But one of the points he made there, which particularly attracts me, is that this was a system-directed, not towards a particular language, but the attempt to make it independent of all source languages. I'd like to link this to a comment I made earlier, that I think when we consider extensions to any given language, we should first consider it in the context of, can this possibly be an extension to all languages by keeping it as independent as possible from the characteristics of any given one? So we can bring all of them along together and minimize this training problem of having to learn separate systems for separate languages.

MODERATOR:

I think this would be a good time to break. We're keeping right on schedule. I think we've set the groundwork for tomorrow's discussion. Reassemble in 15 minutes.

[Break]

MODERATOR:

My choice to moderate this meeting was fairly accidental, but actually was quite appropriate because at home, I carry the banner for standardization, whereas at the CODASYL Committee, I'm usually taking a rather dim view of standardization. The fact is that standardization is a grand, good thing, but can cut off innovation. I think it's what we're talking about a great deal of the time here, is when is standardization too soon. Now, our panel has a good deal to say on both sides of this question. The first speaker is far better known than I am. I've known Joe Cunningham since 1945, and I think a good many of you have known him for a good while. He's going to present the case for standardization in the federal agency. Joe Cunningham.

CUNNINGHAM:

Thanks, Jim. After listening to the comment this morning about the Short-Range Task Force, and now he reminds me of 1945, I'm beginning to feel kind of ancient. I don't know whether this is significant-- Oh, yeah. The cash bar will be in this room rather than in the South American Room. Apparently, somebody got soused over there or something, and they had a little difficulty with it. The latest flash is that the last session was so good, that Bob Curry has just departed to Europe. I thought I got most of it, but I don't know whether he was confused or not.

I couldn't help but pay a lot of attention to Howard Bromberg and several parts of that excellent luncheon talk. But one thing that intrigued me was the notion that he had 10 years ago that if you had an idea, that was the only thing. Then later it got to be the timing of the idea, and then ultimately how to administer it. I'm sure he's probably been getting some news reports out of *Playboy* occasionally, because in a very recent one, that philosophy was borne out to experience. It seems that the folks who operate the Leaning Tower of Pisa have decided to install a clock in it because after all these years, they found out there was no sense if having the inclination if you didn't have the time. [Laughter] I'll tell you, that Short-Range Task Force report in 10 years shows that we've got time.

I couldn't help but pick up something else, though, at this hour. There may be some significance to a little bit of this, but I noticed that Dick Kerrs is very, very objective. He runs into a problem, he says, "Well, if we can, and I think we can," which reminds me of a very old poem that I'm sure several of you have heard of, but I'll bore you with repetition about the teacher who asked the little kids what they wanted to do when they grew up. She called on Herby first, and little Herby stood up in all of his majesty and said, "When I grow up, I want to be a man. I want to go to Japan if I can, and I think I can." And I guess he did. Then he asked Jeanie, and Jeanie got up and said, "When I grow up, I want to be a lady and I want to have a baby if I can, and I think I can." She asked little Howard to get up. Howie got up and said, "When I grow up, I want to be a man, but I don't want to go to Japan. I want to help Jeanie with her plan if I can, and I think I can." [Laughter]

I'm up here talking because of the strange notion that some of your representatives have that the federal government is a monolith, when in fact, as you know, it's the most conglomerate assortment of conglomerate of conglomerates that you can run into. Yet it plays quite an interesting role on us, and if we look at ourselves today, there are a few things going on that I think we ought to bear in mind when we start thinking about where we're going for the future. The federal government has two faces: it has a sovereign face, and it has an administrative face. I suspect that somebody can speak for both sides of these; I can't. But I want to dispose of one in a hell of a hurry because there is much to be said about things that are going on that I think we ought to all be at least cognizant of in addition to procedural/nonprocedural data descriptions. On the sovereign side, as you all know, we have the Congress, who makes our laws. Then we have outfits like the Federal Committee Commission, who is now trying to find out what the interrelationship between that Land of Watsonia and the Ma Bell system they have. We the Patent Office that has a few interesting inputs into our way of life. And we have another one that's going to play a little role in the Small Business Administration. They have quite an interest in what's going on.

I think in these 10 years, one of the things we might realize is that we've grown up to the point where now all these agencies are today paying a lot of attention to the discipline that we think is all our own. Then of course, we have another agency that's down in the Constitution that you call the Department of Justice, who, together with some other companies, are introducing questions and decisions in the courts, who have quite an impact on our way of life. Up until now, maybe some of these agencies haven't played as important a role as outfits such as the National Science Foundation and the grant agencies, who facilitate, many times, the developmental work that we talk about in meetings like this and many others. There's another character around that most of you recognize. He probably has the easiest job in the world. His name is the Controller General. He's the only guy in the government who has the right and the authority to tell you how you should have done it. What's more than that, he has a profound influence, and that influence has sort of the domino effect on our society as a whole, because in 1960 he wrote a report. He said the computer is going to do a lot more than most of these people are talking about, and somebody ought to manage it someplace in the federal government. He suggested the Bureau of the Budget, and the Bureau galloped in and hired a guy who said he had seen a computer once or twice, and assigned him to the supply section. That's where computer policies were going to be made.

Unfortunately, some of the things he did were good, but by 1963, the Controller General had another issue. With all these computers around that we were going to get rid of in two years, and if there's anybody who has his name on more lies to that had effect than I have, I'd like to know. He must've been my boss, who signed the papers after I did. And he said that they never were there, so why don't you buy some, which influenced what the government did for the next few years. Now, I think that in looking at the government universe, you ought to realize that about 60% of our computers are owned. Maybe that's why we have so many second-generation and so few third-generation. Last year he wrote another report, and that's why you ought to consider doing maintenance in a separate package. Within another couple weeks or a month, there's still another report coming out, which may have some more impact on it.

Now, I mentioned these kinds of agencies. Not that we can respond, but we have to look at what they're saying so we can see how other people are looking at us. The Controller General, I said, has a nice job. Let me tell you, he's the only man in the world who gets you to confess. He writes a report and he says 20 things are wrong. He doesn't want to criticize you unnecessarily, so he sends the report to you and says, "Tell me what's wrong with it." So you go through and you're vehement to these kinds of criticism, and you show him eight of the things that are wrong, and you send it back. He takes all eight out, and what have you done? You've signed a confession. [Laughter] It took me a long time before I realized that, and I convinced my boss about 1964 or '65 that we've got to send the report back and say, "Thank you," and we did. We had a hell of a time first getting it out of the Defense Department, where I worked in, and secondly, then keeping the Controller General, who would come around, wanting to know why I didn't say something. I'm not sure now, but I don't think the report was ever published.

On the administrative side of the government, we have a little different situation. In some of the problems that we talk about today, to varying degrees as they existed years ago, were recognized by others. I think that the recognition of some problems of a broad administrative responsibility are best depicted by the first meeting of CODASYL, which is sponsored by Defense and the Charlie Phillips' foresight and willingness to put up a conference room, which were hard to get in the Defense Department in those days. Look what it led to. As I mentioned, in '63 the Controller General had written his report about buy versus lease, which resulted in some legislation that would have set up an agency of the federal government to be a czar, following somewhat the philosophy of the old military, that if you complained enough about the cooking, they'd make you a mess sergeant. If you didn't know how to get computers to work on simple, first-generation systems, then they'd make you the czar of the computer system. 1965, there's a presidential-appointed committee that wrote a report about many of the problems of management in federal government, and I would suggest to you that some of the things they say about standardization in there are still appropriate. Finally, in 1966 we have a wall that sets up some sort of a unified system of operation, with the Bureau of the Budget having responsibility for providing policy guidance and referring squabbles, and the TSA having responsibility for procurement and inventory management and general sense, the Department of Commerce and the National Bureau of Standards having responsibility for technological services and the recommendations to standards, and finally, and most important, the users. The users have their rights protected so that you can go to a certain degree in kibitzing, but you can't go in and take the other guy's wife for him.

Now, this strange notion suggests that the 4,200 computers that the federal government has installed today—which does not include all of those that we pay for; it just includes those that are operated in the government—those 4,200 computers are made by 11 manufacturers. But one or two interesting statistics. No manufacturer provides more than 28% of the computers. This is a startling variation from the national situation. But with it comes all kinds of problems. As we looked at jobs through a pipe in 1960 or 1959 or 1963, and we decided how we were going to handle that particular thing, we lost sight of some of the long-range implications of it. So, in agencies where we have two computers working, as somebody said this morning on a big distinction between scientific and business data processing problems, we have one working around the clock with a four-day backlog, and the other one's sitting there waiting for some work to do, and you can't bridge the two. Explain this to congressman, to the Controller General, or to one of your board of directors, and you have an interesting philosophy. They turn it right back and say, "Well, come on, fellas. This is your technology. Why are you doing this to us?"

As we run into these problems, we find that we become victims of ourselves, and in a way, we haven't had the division, maybe. Or maybe we've had it. And we deliberately solved the problem of the day. I don't say there's anything wrong with it. But let me give you a few interesting figures. You're talking about data division and data banks and data languages and so forth. We look at some of our files of data and find that they're captive to a particular kind of a system. One agency, the Social Security Administration, has 130,000 reels of tape, which are data files. All of these are "active", would have to be converted when they move from one place to another. The Internal Revenue Service has about 110,000, and I'm sure most of us wish they'd lose those occasionally, particularly around April 15. Then we have many, many smaller installations with smaller files, smaller problems, but maybe because they're smaller, more diversified files, there are greater problems of file integration than we have in these larger ones, which are rather dramatic.

Probably, the largest one of all is the one that many of us who were around in the early days were concerned about. Jack Jones, when he worked for the Air Force; Al Ash, who's here today; two or three other folks at the Air Force. We now have in the wholesale logistics operation about 240,000 reels of tape. Those are big numbers, huh? There's a lot of data in there. But there are also some rather serious problems of ever moving from one place to another when you consider that in a procurement sense, the government's ground rules have been, and will be, and presumably always will be while we have an industry rather than a regulated activity, if we continue to have a competitive industry, will always be that there must be competition. When you run into that, the first thing is, who's going to pay the cost of converting from something else? That's not important, I don't think. It doesn't like the Bureau of the Budget guy talking when you say cost isn't important. What is important is where we get the time to do it. No matter how fast you write a reel of tape, it's surprising how long it takes to convert these kinds of records, particularly if there's any degree of dynamicity to them.

For these kinds of reasons and a strange notion that some of our board of directors have that the government have, we inherit 4,000 computers, scattered around, dedicated computers, computers working on general purpose activities, all of which are restricted, to some extend, by the kinds of problems that we start out to solve. Working with COBOL for many years—not technically, obviously—but working with COBOL for many years, one of the things that astounded me was the fact that it didn't catch on. It caught on in places where there were guys that were dedicated, like in the installation. It was under my immediate jurisdiction; had four or five computers. Like many other things, it was a first. We're the only ones to go up in flames. Most recently, the FS would go down in flames if it went anyplace. But when you got into that kind of a problem where there people around who were promoting that the guys would like to use the language that the boss was interested in, and so we got kind of active in it. We found it was very, very helpful; but we found in other areas, we couldn't do it. When we spent millions of dollars to buy computers against the wishes of our senior people, it was not because we wanted to buy them; it was because that by any sense of industrial engineering techniques, you were five years away from reprogramming out of the mess that you had programmed yourself into in machine language and assembly languages. Consistent, if you will, with that other basic premise that in the government, you live in a goldfish bowl, and the competitive procurement is very necessary. Of course, that's one of the reasons I mentioned before why we have such a small, third-generation inventory. On the other hand, in a different set of economic ground rules, we find that figures say that any place from 70% to 90% of the third-generation computers are trying to make like a second-generation computer.

So, we have problems of that type that I think we ought to face up to. There are many technical problems, and it truly isn't my intention to attempt to talk to the technical problems. I'm just talking to the kind of a problem that I get faced when an IA congressman calls me and wants to know why something happened or why something didn't happen. Or when the good congressman, Jack Brooks, calls me, and I think most of you have heard his name. If you have ever met him, he's kind of a country boy that talks with a little accent, and before you know it, the knife is not only in your back, it's through, and you're not sure whether you've been hit or what happened. But he called me one day and he says, "You know, these standards, I'm going to write you a letter." He wrote me a letter and he says, "Standardization is something we need. It's very important now that we standardize on horseshoes. That's about the pace our procedures are working. So maybe we ought to give some thought to standardization before the fact rather than after the fact." So I think it's an interesting challenge. I don't know how to do it, but I think it is an interesting challenge, if we're going to promote the developmental effort that's necessary. Now, how do you decide early in the game that such and such had a file maintenance system that is better than any other, and that's the one we ought to standardize on? I don't know the answer.

One of the reasons that CODASYL has asked for this meeting is to raise some of these issues and get answers out on the table, so hopefully, we can find ways of doing it. Maybe we ought to be careful in what we define that we standardize, and not try to do everything… Of course, we're not getting very far with anything. He wrote a letter in March of 1968, which the then-president of the United States signed and said that starting July 1 of this year, this is a policy. It took almost 12 months before we got the implementation instructions out. Not that the people in the Bureau of Standards weren't smart; they knew how to write letters of implementing instructions. But if you have ever seen foot-dragging in your life, it was the visitors who came in who were raising hell about this thing or that thing or something else. There are some very (and if you'll excuse me, he is still here), very pragmatic problems, profit problems involved when you make decisions of this type. Many of them are not like the decision you have to make when they build that tunnel from England to France, and there was going to be a hell of a note if they both go in the same time. That kind of a decision has to be made, and everything cuts over at midnight or some hour. But some of these we have to make transitionally and go down through the years, so we work in that general direction.

I've just tried to talk a little bit about what happens when somebody decides you have a different kind of a monolith and that you have to integrate these things. When you have problems, you really want to buy another one of these computers, but you've got five of the other kind around. But can't you use the five of the other kind? Unfortunately, you don't answer this kind of a question to some esoteric systems designer. You answer to a stupid official who's interested in economy. I think we are all, at any rate, and so far as the other guy is concerned. Now, we ought to look a little bit to the future. Incidentally, that Brooks letter caused a little controversy, but he called me later and he said, "Don't try to answer it." I said, "I was hoping you'd say that because I don't know how to answer it." He said, "Well, don't try." But maybe it will raise some issues that people ought to give some consideration to.

When we look down the next 10 years, I think a few of these things we ought to give consideration to. What's going to happen from these regulatory agencies and sovereign agencies and result to court decisions? Who knows? I don't. There are chances that the complex that we've accepted the things that we lean and maybe broken up in one way or another, or maybe pulled together into different factions. Maybe we're going to have different problems in managing installations. We don't know. But these are some of the things we have to face, and more importantly, we have the face the criticism and the comment of the Board of Directors, which, in our case, happens to be about 600 people. That's sort of a vicious board on occasion. When you forecast the future, this is very difficult, as we all know. Speaking of forecasting always reminds me of a little bit of a meteorologist; if you don't know what a meteorologist is, he's a guy who can look in a girl's eyes and tell weather.

I listened to the conversation early today, and I can't get away (I'm sure we'll get this back later), I can't get away without telling you that the first boss I worked for in the Bureau of the Budget was a very dynamic guy. He was expendable when Congressman Mills refused to act on the tax increase, and it had some economic impact. When he left, however, for the first and only time in the history of that administration, at any rate, at the farewell party for a non-secretary, but for a guy who has everything except the honors that go with it—he certainly has the power—the entire cabinet turned out. The President and the Vice President. Some guy and got up and said of Charlie Schulz, who is the man I'm speaking of, that never again in the history of the United States will a man appear before Congressman Mills and make the following statement: "I know you believe you understand what you think I said, but I'm not sure that you realize that what you heard is not what I meant." Somehow, that may fit the meeting we're having here. Thank you. [Applause]

MODERATOR:

I think you mostly know, Charlie Phillips had a great deal to do with COBOL getting its start in being the prime mover and keeping the CODASYL Committee active for many years. He, after numerous years in the Department of Defense, switched to private industry as executive director of BEMA, and in that capacity, he is also chairman of X3. Now, the X3 Committee of USASI has standardization as one of its main activities. I'm not sure about this. It seems to me that perhaps he anticipated Howard Bromberg's remarks about there being some case of pregnancy before marriage, and the standardization effort is sort of an effort to do something about that situation. At any rate, Charlie Phillips now is going to present the case for standardization in general and standardization of COBOL in particular.

PHILLIPS:

Correction: My job is not Chairman of X3. I'm Director of the Data Processing Group in BEMA. I was elected Chairman of X3 before I ever left Defense, and I've continued in that capacity since. I want to apologize for my bad voice. I hope to make it short, though; I won't bore you too long.

What I'd like to talk about today is the X3 operation. Give you some of the history of it. In that sense, it might better have been done this morning. I'm very grateful to Dick Kerrs and also to Bob Bemer for some of the things that they covered, which will make possible for me to skip them and to cut my talk a little bit short.

On the same day that the Executive Committee met here in Washington in the Southern Railway Building in January, the ASA, now USASI, called a general conference in New York for the purpose of discussing the need for a standards operation in the field of computers and information processing, onto which grew the X3 Committee. Actually, the impetus behind that didn't grow here in the States. It came out of an international conference that had been held in New York the preceding year, which the Swedish representatives had suggested the need for such a program, and such a program was organized internationally. Therefore, we are faced with the need for providing a national input to it. Bob Beemer, who is scheduled to be with us on this January meeting here in New York, didn't show up; called us, quite excited, to tell us about this general conference that had been called, unknown to most of us at the time. But X3 grew out of that activity. There are two other standards committees in USASI that were founded at the same time: X4 on office machines, and another one that has since passed away, X6, which had the electrical characteristics of computers, as distinguished from the logical characteristics that was assigned to X3. Over a period of a couple of years, it was found that this division was completely impractical and artificial, so X6 went its own way.

Beemer was asked to sponsor this X3 Committee, and they held their first meeting in the spring of 1960, which means that the X3 Committee is approximately one year younger than CODASYL. I've been asked at times whether had X3 been in existence at the time that the idea group from Moore approached us in Defense to see if we'd kick this thing off, whether Defense would have done that at the time, I really don't know. I think you ought to ask people like Grace Hopper, Mary Hollis, Sal Goren [?], or some of the others in this idea group as to whether they would have proposed this. If I was asked on whether the results would have been the same, I think I'd have to say no. I think that through the effort being lodged where it was in CODASYL, we would have made much better progress than we would had it been assigned to X3 at that point. I'm basing this on experience.

Back to X3, when the Committee was first organized, it had approximately 30-member bodies, which has since grown to 44. In case you're not familiar with it, and to give you a general feeling for the diverse interests that are represented on X3, I think I might quickly tell you who the interested numbers of X3 are. The Administrative Management Society of the Association for Computing Machinery, the Data Processing Management Association, the Electronic Industry Association, Industrial Communications Association, the Institute of Electrical and Electronic Engineers, the Joint Users Group, the National Bureau of Standards, Systems and Procedures Association, the Telephone Group, the American Institute of Public Accountants, the American Society of Mechanical Engineers, and the Association for Educational Data Systems are going to make up what we call the General Interest Group on X3. There are 13 members in that group.

The users are comprised of the Air Transport Association, the American Bankers Association, the American Gas and Edison Electric Institute, the American Newspaper Publishers, the American Petroleum Institute, the American Librarian Association, Association of American Railroads, the Printing Industries of American, Department of Defense, General Services, Insurance Accounting and Statistical, Life Office Management, the National Machine Tool Builders Association, the National Retail Merchants, and Scientific Apparatus Makers. Fifty in this group. The producers or the manufacturers are drawn from the membership of the Data Processing Group in BEMA.

We have 24-member companies. Each year enough of them volunteer, usually, to balance off the smaller of the other two groups. We never have more manufacturers represented on X3 than in the smaller of the two other groups, so the users and the general interest members of X3 outnumber the manufacturers two to one.

In the original X3 organization, there were five subcommittees: X3.1 on optical character recognition, X3.2 on coded character sets and input/output, X3.3 on data transmission, X3.4 on common programming languages, and X3.5 on vocabulary. Obviously, the one that we are most directly concerned with in this group is the X3.4 on common program languages. Since then, there have been three additional committees added: X3.6 on problem definition and analysis, and there some overlap at least contended between that group and the X3.4; X3.7 on magnetic character recognition; X3.8 on data elements and their coded representation, in which this group also has some interest.

Back to the X3.4 and the program languages group, one of the things that was mentioned earlier is the developmental efforts. I think in the previous panel there was a question posed as to whether additions, extensions, or improvements in COBOL should be done by some activity other than CODASYL. I think I should make it clear that at the present time, we look to CODASYL for developments in the COBOL language. There is no requirement that do not do development work in X3. In fact, we have the responsibility under USASI procedures to maintain every standard that is approved. So if we do not get the support and the work from the CODASYL group, we will have to undertake to do it ourselves at the appropriate time. This, I don't think we want to do, certainly not in our present planning. Bob Bemer also covered some of the other development areas in programming languages, so I can skip that.

We have, through X3, approved two standards in the field of programming languages: X3.9, FORTRAN, and X3.10, basic FORTRAN, were approved back in 1966. Work is currently going on, as I think was reported to you, to clear up some ambiguities and improve the FORTRAN language. In August of this last year, COBOL was approved as X3.23, and I think there's something that probably is not known. If it were, we might not have been so highly criticized in the recent issue of *Computer World*, but be that as it may. At the time that we took the approval action in X3 back in July, as I recall, or even earlier last year, we did not have the document. We were basing it upon the subsidies of CIB 9, and all members of X3 had CIB 9. We were assured that the X3.44 Committee would not deviate from the substance of that; that it would be reviewed by CODASYL to be sure that the substance was the same, that all the changes that were made in the standard itself would be editorial. That promise was, of course, kept. But the document itself was developed through the good offices of the Bureau of Standards. We did not, even at the time that we approved it at the Information Processing System Standards Board in August, have a document, nor did we expect one until late in '68. We actually took delivery on the standard from MBS on the 12th of December. We had expected, at that point in time, that it would be a couple or three months before it could be published.

In fact, in about mid-year or early fall, we began negotiations with MBS to have the COBOL standard published by the federal government. We did this because of some problems that we foresaw in the field of copyright. It's the practice of USASI to copyright all their standards. We had pointed out that this was completely impractical and improper in the case of COBOL when it was in the public domain, but we got nowhere. USASI were firm, and definitely turned us down on any compromise arrangement. Therefore, we went to MBS. Initially it did appear that the government could do this, could publish the COBOL document, which would have given us a document of substantially lesser cost than it is today.

However, when we got up close to the publication date, they were unable to do what they initially thought they could, and we were not able to have it published, as we had originally planned, through the federal government. We therefore went back to USASI, and we got them to agree to leave the so-called acknowledgement section in the standard, which, in essence, negates the copyright because it authorizes anyone that wishes to use any part of the specification, in whole or in part, as long as they're using it for some specific purpose. It does prohibit anyone from republishing the standard complete with cover and title, which I don't think is too serious a problem.

I believe that Bob Bemer did mention PL/I. However, there are some developments in that field that might not have been clearly brought up. We had an ad hoc committee, as I think was reported in X3.4, which recommended the development, in a subcommittee of X3.4, of the PL/I language, making it ready for standardization. This proposal was brought before the X3 Committee in October and was tabled. At the January meeting in Phoenix, it was on the table and debated at some length. The final result was that we agreed to find what we call a composite language development group, which is to work in concert with IFIPS 2 and ECMA 10, and doing further development work. I think I should report, too, that while we have 17 or 18 candidates for membership on this committee, we still do not have a chairman. As soon as we find a chairman, we'll move ahead.

You might be interested in the reorganization of X3, which has been discussed from time to time, and which in spite of rumor, is actually proceeding. The Systems Advisory Committee, which was appointed something over two years ago to deal with interrelationship of standards and families of standards, when they began their task, concluded that before they could really move ahead, X3 needed reorganization. They presented to X3 a year ago, approximately, a reorganization plan, the effectiveness of which was to separate the planning function from the mind work. In other words, it would have two staff committees, and then the balance of the operation would be a line operation. On the staff side, they continued the Interrelation Advisory, which we had had for some time, plus a new committee, which was called the Planning and Requirements Committee. This is the group that would review scopes, would review proposed projects, would review the finally proposed standard to see if it conformed with the original basis upon which it was referred, and would also do general long-range planning. It would provide it for, as a part of the line function, a Standards Staying Committee composed of the chairman of the respective subcommittees. SAC's proposal also suggested that we completely eliminate the first level of subcommittees and go directly from X3 to the task groups. This particular part of the recommendation has not been accepted at this point, but other than that, the proposal has been accepted by all persons that are involved. We have a Standards Planning and Requirements Committee called SPARC, which will hold its first meeting in conjunction with the final meeting of SAC. That's scheduled within the next two months. I believe that we will probably see the full organization of the committee going back to the recommendations of SAC. Possibly not completed during this year, but certainly well along by the end of the year. We're having difficulty finding people that are willing to take on the responsibilities of discipline coordinator, as they call them, which would be a grouping into three areas of the Hardware Group of Standards, the Software Group of Standards, and the Systems Group of Standards Activities.

About a year ago, also, a committee was established to rewrite the procedures of the X3 Committee. This group undertook also to put in writing the proposals of the Systems Advisory Committee on reorganization. Their work has been completed. Their proposed manual has been cleared and released, and it'll be available within the next couple of weeks.

I think I might briefly talk about the results that we've achieved in some nine years of effort. We have some 28 standards, which seems like a pretty poor result for the amount of time and effort that's gone into it. We attempted to price it out, and I think that these standards cost us about a million dollars apiece. Now, whether they're worth that or not, I don't know. But in 1968, 62 different committees that are a part of the X3 structure held some 248 meetings. This actually represented 390 meeting days. Participating in these meetings were 3,461 people, who put in a total of 18,500 days or something in excess of 71 man-years. If we put a rough price tag on this, and I assure you it's rough, this means that we had over $3,400,000 tied up in actual meeting time. If you would add to that another $2.3 million for preparation work for these meetings, you'd get a tab of over $5.7 million, of which the producers contributed 49.8%; the users, 24.8%; general interest, 4.7%; and the government, 20.7%.

That very well gives you a picture of what we've done, what we've achieved, what we're looking forward to doing in terms of reorganization. Thank you. [Applause]

MODERATOR:

I apologize. I thought Charlie was going to give us the case for standardization of COBOL. As far as I could see, the case is, more or less, made opposite way—in particular, in regard to this thing they call PL/I, which I assume stands for Public Law Number One. Is that correct? Now, some of us have expressed the view that a great deal more might be accomplished if there were some lessening of the duplication of effort that seems to take place here between the groups who are with the prime motive of development are doing some standardization, and the groups who are with the prime motive of standardization are doing a lot of development work. Or, let's say, if they're not doing it, they have the people who have tendencies in that direction. I question, and I think Warren Simmons, the next speaker, has also raised the question of whether there is an opportunity for reorganization of this effort to accomplish more with the same total effort. Warren Simmons of U.S. Steel.

SIMMONS:

Thank you, Jim. I was very interested in Charlie's reference to the dollar cost of the effort last year in USASI, and in particular with the comments, that Jack Jones or someone else made earlier this morning about the fact that we haven't spent any money yet in CODASYL. The $20 was the first time we've asked for that. I think the CODASYL investment is somewhat like the cost of multiprogramming: it's all buried in the operating system. [Laughter] The questions that were posed in the questionnaire for this session are, "How can X3 and CODASYL best interact?" and "What role should the federal government play in developing standards for the data processing community?" and "How can other countries and associations participate?" and "Should users participate in the development of hardware standards?" As was clearly stated the first thing this morning, all of our discussions are personal opinions, not CODASYL opinions. Mine, in particular, is an effort to give us something to talk about following these two gentlemen's speeches. And probably clear into the cash bar for the evening. Why not?

CODASYL is engaged in trying to identify problems that can be solved by the design of data systems languages. It's been said several times. And then trying to solve these problems through cooperation of those members. The marketplace, however, plays a very healthy role in this interaction. CODASYL pioneered the standardization of COBOL by creating a required elective. One of the fondest memories I have, Howard, is the game we played here in Washington on voting on each element as to whether it required an elective, and your part of that particular activity was one of the minority hardware man with variability. It was always voted elective. Later, people probably don't realize that COBOL Committee identified a minimum language level, and then dropped this in favor of the cooperative effort with X3, ECMA, and the Japanese, and produced the COBOL standard. So I think we are identified with the standardization effort, where very solidly so. But for many reasons, believe that the CODASYL mode of operation has to change to keep peace with its goals and to work with all organizations interested in data systems languages, regardless of the language. People have been avoiding using the words "PL/I" or using it mostly in a humorous way. PL/I is one of the data systems languages that we must deal with.

Each of these heretofore independent entities must contribute their individual expertise to the mutual advantage of all. My position, as one possible alternative, is that all common user development should be conducted under a strengthened CODASYL organization.

For now, the overall data systems languages situation has developed into a situation that reminds me of a cat with diarrhea: it has four friends digging holes, four more covering up for him, and you have four more scouring for new places to solve the problem. Now, how can all of the different factions get together? I feel that there is a very unifying common objective which supports such a move: the development of a common database of data description language. The result of current X3 and Database Task Group efforts, when combined, could be a common database for procedural and nonprocedural languages. The benefits would allow users to interchange data between company functions whose language requirements differ; allow exchange of data in machine-readable form, such as order shipments and reporting information, and much more; provide a well-defined interface between the many data system languages that exist; show the way to the identification of the most useful way to provide language capability; remove true redundancy where required; and finally, provide a uniform procedure, both to and from participants in the CODASYL organization, to accomplish the greatest good for all.

I propose that the following steps be taken by CODASYL to bring this about: One, offer to provide consultation and coordination for all software development activities. Two, request some of the major federally sponsored data system language efforts to extend our participation as a means of staffing the same efforts in CODASYL. This would permit the consultation and coordination mentioned in Item One and Eight in the development of data description languages. Offer to manage the software development activities of X3; bring them into CODASYL. I've heard that there are none. I feel that the activities that they have in this area aren't properly classified as developmental activities, however. Organize all language development activities, thus formed as standing subcommittees under a reorganized Programming Language Committee. Then establish that members are representatives of associations or represent a class of users or vendors. That is, hardware companies are representative of BEMA; that software companies represent the Association of Independent Software Companies or other associations. Users represent users associations. The federal government participants would each represent all agencies in their activities. If X3 and user associations, the federal government participants, and the several pertinent data systems languages, as well as the organizations such as ECMA, the Canadian government, and the Japanese, were to cooperate in such a change in the organization of CODASYL, the benefits would be the concentration of user and vendor participation and development. Apply greater total pressure to solving the problem. Improve communication between all members. Provide greater numbers of people for development who have more resources at their disposal. Establish a single procedure for development of solutions to common problems. Finally, permit CODASYL to sponsor the standardization effort when development reaches a good level of acceptance. Jim? [Applause]

MODERATOR:

So there you have it. The views of the committee are not exactly unanimous on these subjects, as you can see. You'll have certain questions that have been posed here. Who would like to comment on these or any others? Mr. Beemer?

BEEMER:

A good opportunity happened as I was chairman of the Systems Advisory Committee (SAC). I'll say it once more to many people: This is the most satisfactory committee I've ever worked on. We never took a single vote. We always had a consensus that came through this. This has some mighty fine people on it—Bob Brown from the Department of Defense, Mark Reneger [?], Doug Ross, Bob Ralphie [?], and Eric Kramer [?], Lou Robinson from IBM. What we proposed was the integrated package: the staff function, as has been accepted; and also the line function, which has not been accepted by the Processing Group of BEMA. Now, I feel that in not accepting this line function as we proposed it, we will have probably a little bit more than a camel. I say a double camel, a Siamese twin camel.

The reason for that is that I don't know if all you people here have seen the ball-bouncing that goes on between X3.4, X3.4.4, and X3.4Q, and they agreed on here they'd bounce up one level and said, "No, no, go on back up here." When it finally passes this level, it goes up to X3.4. They said, "No, no, go back there," and it goes down again. What we propose is one level of technical agreement. With X3, all these American Petroleum and stuff, as the administrative thing to see that that technical agreement was properly reached and was a valid agreement. Now, we did have the ulterior motive, which was never stated in this organization, but this type of having a functional responsibility—that there, in fact, would be a COBOL Committee, that there, in fact, would be an I/O Interface Committee or a Disk Pack Committee that was single—would allow X3 to deputize the standardization work. This goes very much with what you're saying; if you can develop but at the same time keep it in your back pocket, you've got to do the standardization work, too, and it may come along like this, maybe. Then we could avoid a great deal of this duplication that does go on, and deputize CODASYL for a great number of these functions.

That will be impossible unless we do two things: First, get that X3 reorganization on the line level, as we agreed. Secondly, revise the CODASYL publication procedures so that when you do propose a standard of X3 for administrative approval, it has received the necessary public distribution and comment required by USASI. I think this is the particular case with the data communication language that was recently adopted. This could not be in a standard because it got seen by very, very few people. It did not get published, for example, in *Communications of the ACM* or *Data Matrix* [?] or something like this. This would be very vital for it to be a part of a standard COBOL. But I do think that if we change the publication procedures and the reorganization of CODASYL and allow X3 reorganization to deputize, we might achieve some of what Warren wants.

PHILLIPS:

Can I respond, please? There is general misunderstanding, I believe, about publication policy being a USASI requirement. That is not so. We imposed this publication requirement upon ourselves in X3 back in 1961, as I recall. We don't have to observe it. We have observed it; we believe it's in the best interests of the data processing community to distribute the standards, give the community an opportunity to review them, see whether they're logical, and submit comment before we vote on them. There is some flexibility in it. We normally do not vote on a standard until it has been published for a period of approximately six months. We can, and I think we have, done some things to shorten this period in the sense that we can have a concurrent ballot during the time that the publication period is running. We are trying to speed it up, and I think we've achieved some improvement. But publication policy is one that we could dispense with, if we wish to, at the next meeting of X3.

MODERATOR:

Howard?

BROMBERG:

I think we'll reorganize and deputize CODASYL, so he's running after the Simmons, when Simmons is going to take him over and put him out of business, which is Warren's idea, and then he's going to eliminate the X3 requirement by doing everything within CODASYL. It appears that what they both recognized is that each was doing part of the other, so each could, if you have the sole responsibility, undertake in a much more economical, and possibly elegant, fashion to do the whole job. I think this is a subset or a part of the problem, which is, can committees function in today's complex environment? Perhaps we should really look around to see how CODASYL as an organization can fund, for example, those kinds of activities that are near and dear to its heart to those organizations, universities, or what have you, who can do a job quicker, faster, and better. And X3 can do the same thing with respect to standardization. It has been the argument they are duplicating effort, they are taking a lot of the problems, but you're not approaching them with a solution to the problems that are out there.

MODERATOR:

I don't believe that Warren had in mind elimination of either one. Certainly, I did not. I do think that there's a possibility of combination. I think that's what you're indicating. Combination under a unified direction. This is what, it seems to me, is most needed.

BROMBERG:

I think that's possible. You have two huge organizations, and the inertia of organization itself is such that you can't stop them.

MODERATOR:

Well, they say the federal government is impossible, too, so I don't know. What other comments do we have? Dick?

KERRS:

I can't help but fear the inertia of these organizations also. It sort of a touch-and-go ???. You can't do anything wrong very quickly, also. I fear a very swift, quickly paced distance between development and standardization. I fear that it might simply result in more unimplemented standards. I don't know. Maybe my fears are unfounded, but they're there.

MODERATOR:

Over here.

BUSHED:

Anheim Busha [?] from MIT. I sort of feel about the data descriptive language work there hasn't been enough ground work for approaching the problem of standardization in this area. I haven't yet seen a really a clear, descriptive presentation like what kind of facilities could exist, but even the perimeters have not been defined. I think the various members and various organizations including the US, which I belong to a university, and different manufacturers can start thinking in these terms to develop the perimeter, and then the task can come towards the USASI task group or something for the standardization. I think the ad hoc committee for X3, for which I was working on the subgroup, did decide on such a thing that really the time is not right to talk about standardizing on a data descriptive language.

MODERATOR:

Down here.

MALE:

Could we at least clarify the lingo here or direct suggestion that I've heard more than once about CODASYL being recognized.

[CODASYL Disc 6]

OLLIE:

--with 15 years of brilliant hindsight. I think one of the mistakes that we've made in the computing community and the development of languages is to allow the definition of the data to be stored with the program rather than with the data. I think the founding fathers and mothers of COBOL at least recognize that the data definition should be in some particular place on its own rather than scattered throughout the language, the processors, the program. I'd like to recognize one of the founding mothers, Grace Hopper.

HOPPER:

I have to argue with you on that. The original concept [inaudible] the company or community to select their ??? and to find their data ??? ??? ??? in the library. All programmers would be [inaudible] programs. In this point of view, I think our data [inaudible] our languages. This was the original idea [inaudible] within the community, within the [inaudible]. We didn't mention [inaudible].

OLLIE:

I stand corrected, and thank you very much for that comment. Mary again, please. Another founding mother. [Laughs]

HOLLIS:

These mothers do get around. Actually, it has been rather appalling, and I think I would like to cite Bromberg's statements that the idea is good, and the timing can be good, but it takes a little bit more. Essentially, what I'm saying is it is not sufficient to come up with these definitions and to actually specify this, but you also have to ride heard to see that the people do it, and do it properly.

OLLIE:

Dr. Hopper again, please.

HOPPER:

You must have the idea and enough times tell people about it, but they don't start doing anything. You have to charge and take the first step. You can be wrong. And I will urge all of these committees and all of these [inaudible] because there may be some time sitting there and we can hurry them up. [Inaudible].

OLLIE:

Yeah, the question is where the stage to shout from? Any other comments, please? Yes, the gentleman from Chrysler.

FOGAL:

Dan Fogal. I was kind of bothered by the fact that in the beginning you define the four levels. The one that I fix upon is your Level A, the conceptual level, because this is the place where we've left out everything that needs to be said. I'll tie that into—I think John Gosin [?] said this; there are so many versions of what various people said here I'm getting a little confused. But I think it's attributed to John Gosin as one of the points in the problems in transferring the thing is that you can't talk about his kind of thing in your environment. There is no correspondence of nouns between the two nations, if you want to look at it that way. This is, I think, one of the biggest problems, and then I look at the questionnaire, though, and all the questions have to do with how we shall play with the code and transferring from machine to machine. Basically, the conceptual differences have little to do with machine to machine or with hardware to hardware or data communication. They're from one conceptual framework to another. And once again, I'm back to the same thing. That's the part that bothers me, and I'm kind of mystified on how that somehow escapes from all the questions.

OLLIE:

Bill again.

MALE:

I'll comment on that. I think many of the questions were phrase in the context of a logical description only. Because this is the one where used do and the one that we're most familiar with. But there are problems associated with the other levels of description that the questions don't really adequately reflect. The problem of data translation, I think, really cuts across all levels of description. I don't think you can confine the problem to any one level.

OLLIE:

I think I'd exempt the Level A there, Bill. John.

JOHN:

Let me postulate that by analogy. One way to establish an on-call is to freeze on some machine language. In theory, it may seem that you can't translate one data structure into another. I suspect you can. Read Volume One of Neu's [?] tome on how he maps tables into trees and back again. I suspect that with most of the systems we have around, we can find some useful mapping. We may have to have conventions to avoid the natural ambiguity the clever programmers would put in. But there ought to be a way to sweep a table and sweep a tree and map them one per one. I suspect this is true of anything, which is roughly equivalent to a Turing machine or whatever the logical guise of that in proof. If it can't be done, they'll quickly extend that language to do it, or they don't need to do it anyway, so it isn't a problem. I suspect that that's another of these big red herrings that's being trailed across my path by the theorists who tell me that the perfect solution is impossible.

OLLIE:

Do we have further comments here? Fine. Well, before closing this session, I would like to ask all of those present to try to answer the questions in the questionnaire on this session because your answers will be of great guidance to the CODASYL Executive Committee. I would also ask you to feel quite free to give narrative answers on the sheets provided, because this is one way of expressing your opinion. We tried to get encoded answers because if you can give them, then they're useful, but we recognize that some of these things are a little bit too diffuse for that purpose. So you are encouraged to try to complete your questionnaire and return it to the young lady at the registration desk. The next session will begin at 10:45 promptly. That will be the session on database management. Thank you very much. [Applause]

[Break]

MODERATOR:

The purpose of this panel is to discuss database management again more thoroughly. As you know, we have six speakers. We have allocated two hours of time to this discussion. I'm sure that we will bring in data description languages into this discussion as we have into every other panel. The members of the panel on my left: Tax Metaxides of Bell Laboratories; Charlie Bachman of GE; Chuck Greenberger of Esso Mathematics and Systems; Bill Ollie, as you well know, of RCA; Dick Kerrs from NCR.

Having come this far, and I don't know how far we really have come in the area of database management, the question, of course, to be asked is, "What should we do now, and who should do it?" This is a general question, but it certainly applies to this particular area about business, probably to everything we've discussed at this meeting. Yesterday, someone wisely suggested—at least wisely in my view—that we must have an appraisal of the rapidly changing environment to establish the objectives for the future and to plan intelligently to achieve these objectives. It wasn't worded exactly in this way, but this is the way I read it. When I look at this problem, I'm not thinking in terms of objectives of the systems programmer; I'm thinking of objectives of business, industry, and the government, and in terms of that they consider their objectives. After all, we're in the service business.

A worthwhile objective really—I think you will agree—must reflect the future requirements and not the requirements of the systems programmer. It must reflect the requirements of the business. Ideally, this calls for gathering some information, which is very difficult to gather. It calls for forecasting the requirements of management in terms of both their information requirements and in terms of administrative procedures. Incidentally, we haven't done much of a job in really defining what information is needed for planning and control that must be put into management information systems, so we build these fine tools, but we throw in a large of garbage and we don't really know if it's any value to them. So there's a big job to be done here. We're fine-tuning on a rough fence. We need to know something about, I think Greg Dillon mentioned yesterday about the manpower supply and the level of competence of the people who are going to be employing these tools. So we need an outlook in this area. We need to know something about how the technology is going, and I think this is an area where we have better insight than perhaps anyone else. We need appraisal of the future opportunities. In other words, what are we going to do with all of these things? I, for one, feel that this whole area of management information is very much tied in with the database development that we're talking about here. Unless we can achieve some independence, we can't hope to have very flexible, large management information systems. We need an evaluation of the opposition to the achievement of our objectives, and there are many, many oppositions that can be considered. The burden of conversion being one; competing organizations to do the work, and so forth. And finally, we need to ask the question of ourselves, "What unique capabilities does CODASYL have, if any, to participate in this area?"

I think this you'll recognize as the conventional methodology for considering any new business venture. Database management seems to be no exception. We should look at this as a new business venture. Seldom in real life, however, can we find all of the information needed to answer these questions as we would like. Still, we have to make decisions and make decisions with the uncertainty that's around the problem in view of the deficiency of information. I would hope that the decisions that grow out of our discussions and the conclusions that are reached by the CODASYL group, as well as by other groups who have an interest in these areas, keep in mind that we're trying to put a new product on the market, and we want to put the product on the market that has a reasonable expectation of success. Not technical success, but financial success. I believe you will agree that we must remember the ultimate use of the software we discuss, as well as where the source of benefits that come from that must cover the cost we incur. Except for the service industry, there's no revenue associated with what we do; just cost.

Hopefully, the feedback we get from the talks of this panel and the resulting discussions will serve to guide the future programs of CODASYL and otherwise, and other groups if they're the more appropriate group to handle the work we're considering. I hope that this leads to something that industry, business, and the government truly need and truly can use to advantage.

I'd like to call on Ollie now, since he has some groundwork. We met this morning, incidentally, at breakfast to decide what order we should use in discussing these topics, and I think the consensus reads that we disagree on essentially everything. Ollie?

OLLIE:

Sorry, it's me again. I'd like to take the questions that are posed in the outline as the outline for my ten minutes. In database management development, I think I identified two clear paths. There's the path in which one takes COBOL, or some similar procedural language, and embeds in it database-handling capability. This is instance by such systems as IDS, IMS, DM-1, and things of that ilk. And there's the other "go it alone" type of approach, and we're really talking about both these here in this series of questions. The first question is, "Should COBOL be extended to handle databases more effectively?" Now, I come to you as an alumnus of the Database Task Group, and I think I flunked the course. I was a member of the Database Task Group during the period from May 1967 through to January 1968, which meant eight meetings, each lasting a week. I had the pleasure of the hacking a lot of these questions over with the members of that committee. It's a very illuminating, very educating experience.

I think we have to attempt to extend COBOL not in the IMS way, which is just sort of, preserve the language, preserve the compiler, but do everything through the core statement. But extending the actual language, I think that attempt to make the extension has to be made. If we don't try to make it, we'll never know whether it would be successful or not. I think that in anything of this nature is a four-stage business. First of all, you try to develop the language and the specifications, and that's where the Database Task Group is now. Then the checks and balances are thrown in on the parent committee. In this case, the Programming Language Committee gets in the act and tries to understand things, and I think that is going to be, and should be, a reasonably lengthy procedure. Then we get to the process of implementing the thing. As an implementer of systems, I have a very biased view on thing. I think the people that define systems jolly well ought to be made to implement them and locked up until they've finished. It would probably settle them down a bit. So that's the next stage, and then after it's implemented, you have the acceptance phase: will it be accepted? I think we can clear the political problems out of the way with the backing that CODASYL has. I think that if CODASYL gets something that far, then I think it'll be politically acceptable. So my answer to 5.1 is yes, and that gives me an opportunity to answer 5.2.

There's a game that we used to play, which was estimating the number of atoms in the universe. I'd like to suggest a new game estimating the number of bits of stored data in the universe. It's awful large, and I don't think that you can ignore it. I think that any extended system cannot ignore all that data that is already stored. Joe Cunningham told us yesterday of all those enormous files tucked away in this fair city, and you can't ignore them. Your new COBOL—let me coin that phrase—has to be able to handle the existing data files.

Okay, I think of the one most important question is 5.3, which really could be paraphrased, "Which way is COBOL going?" Which way is it going, which way should it be going. It's my reading, after having served on the Database Task Group, that the way they're going is in the direction of the systems programmer. Let me refer you back very rapidly to the parenthetic remark about systems programmers and the outline. A systems programmer who may use the system in the implementation of a tailored database system, as opposed to a generalized database system such as GIS or Mark IV. They're going to disagree; I know they are. I think that the way they're going, they're developing a tool for this type of individual. Certainly not a nonprogrammer. Originally, 10 years ago, we had a breed of cat called a programmer, and we developed COBOL for nonprogrammers. We now have a breed of cat called a COBOL programmer, and we need systems for nonprogrammers. So, should the extensions make COBOL a tool for-- The way they're going, I feel, is with the systems programmer. I would like to see them go in the direction of the database administrator. What this implies, and particularly in a real-time environment, is an element of privileged operation where certain functions in the language are privileged, and that only this person called the database administrator may use those functions when he does certain things to the data, which not every Tom, Dick, and Harry at a terminal is allowed to do. I argued this when I was on the committee, and I'm not quite sure where they stand. They've got a chance to respond to that.

An applications programmer—should the extensions make COBOL a tool for an applications programmer? That is the kind of fellow that we now know as a COBOL programmer that the DPMA turns out with their CDP certificate. Obviously, if you enhanced COBOL, you still have COBOL there as is today. Hopefully, it is still a tool for that individual. Now, you may put in the bells and whistles on the tool, growing the tool, and I'm not sure about that. I think that's a thing that one has to be very careful of. Should the extensions make COBOL a tool for a nonprogrammer? Mainly a person that does not understand programming as we conventionally identify it? A person that doesn't want to, and a person that shouldn't have to. By enhancing or extending COBOL, you are not going to take a count of this nonprogrammer. I think the only way you're going to get him is to distill, not to extend. Extract the meaningful parts that he needs, and call these-- Let me coin a phrase. I hate to, after listening to John and Jean, but say "nonprocedural COBOL", something like that. You've got to have a handle. You can't do everything. Okay, that's 5.3. I think it's a very important question. I remember, quite distinctly, in some fair city or other, discussing this problem for days on the Database Task Group. Who are we aiming at? Who is the Database Task Group aiming at?

5.4. Should the extended system address itself to problems of database security and privacy? Security—I sort of regret seeing this word here, although I probably wrote it. Security. Do we mean authority to access, authority to change? That's added together. Write parenthesis. Or, do we mean reliability? I think we mean both, and I think the answer is yes. You've got to address itself to that problem. Now, whether the Database Task Group should do it or whether Dick Kerrs should inaugurate another task group to address those problems, I don't know. That's his problem. But I think it's an extremely important problem. I don't know whether it should be subsumed in the web for the Database Task Group. I'll let them argue it back and forth. Thank you, Charlie. Make a note.

5.5. This comes a little bit closer to the work of the Systems Committee, whose flag I bear on this instance. Should a new common language be developed to provide a database interface for the following: database administrator, and this poor person, the nonprogrammer? Should a new common language be developed? That's a very major question because we have, now, a vast proliferation of generalized database management systems very much as we had in 1957 and '58. They're coming out all over the place. The situation is different in 1969 from what it was in 1958. We now have software suppliers. The industry is 10, 20 times as large. It's a much bigger, and, I'm afraid, a much more confused world, although they probably, in those days, thought it was confused enough. But should we have a common language? I feel we should. I feel we should set this as one of our goals. It was one of the goals that was set for the committees, which became the Systems Committee 10 years ago. As I said yesterday, we haven't met it. The question is, should we aim at it? I feel the answer is yes, we should aim at it, and we should aim at it on these two levels: the database and the nonprogrammer. The functions which will enable somebody in a user environment—I'll put a manufacturer's hat on now—customer environment, somewhat enable them to get a new database application on the air rather more quickly. You need somebody called a database administrator; somebody who knows certain parts of the system that other people do not need to know. This person should be able to define certain things about the data structure, certain things about the storage structure, essentially using the type of data definition language, which we have just finished discussing in the previous section. Should define the data. That definition of the data should be stored with the data, or with some catalog of data definitions from which it can be invoked.

The nonprogrammer. Who is the nonprogrammer today? He's certainly not the fellow that likes COBOL programming. Is he the manager with his terminal in his office? Who is he? I think we can push the interface between the man and the machine further out towards the manager with the terminal in his office. I think this is borne out by the fact that we have so many systems being developed in the industry today. I'm developing one; I don't mind telling you, although I'm not giving a sales pitch here. It's a very interesting experience. Just how simple can you make this thing to use, how much power can you give it. You're dealing with some very complex tradeoffs. I think we will have—I'm hypothesizing here—a spectrum of nonprogrammers. I call them specified users so that they don't get confused with programmers. I think we'll have a spectrum of specified users. The type of person that understands the simple end of the specified user spectrum, who can ask a simple question of a simple data structure. And then the chap with a Ph.D. in logic that can ask a complex question of a complex data structure, and believe me, they exist. These things are not going to go away. Complex data is complex, and by simplifying it, you may think you're simplifying something, but you're losing something. So yes, I think we need this new common language.

That brings me down to 5.6. If there is a need, can the necessary simplicity for nonprogrammers be achieved? Well, I've sort of addressed this question already. I think it can be achieved. I think a lot depends on how the thing is presented, and this is probably something that was discussed 10 years ago when COBOL was presented and you have levels of COBOL. You had the basic thing, and then the different levels above that. It's the simple thing for the simple soul and the complex thing for the complex soul. The presentation of your language, I think, is important. I hope that we, the computer community, have learned something in the last 10 years about presenting and introducing languages of this nature.

Should such a new language be able to handle existing COBOL data files? Yes. It's the same question as Question 5.2. My answer was yes there, and my answer is yes here. I'll put on my manufacturer's hat again. You cannot ignore the data that's already there and stored. You have to have some way of defining it as it is stored, and we're working on that. But you've got to be able to handle it. It's very difficult to handle the universe of stored data. That's a frightening problem.

Okay, let me just back up to 5.5, the problem with the new common language, and let me go back to my Systems Committee hat. Systems Committee has surveyed these nine systems. As I said yesterday, I think we've learned quite a lot in the process. We've learned that we have some more to learn. Our next step is to study these and other systems some more to try to build a basis. You'll read this; those of you may have done so already. It's somewhere in the introduction. This report is intended to be a basis for further developmental and theoretical work within CODASYL, or without, or outside of people so which. I think we have to go a step further. We have to study the problem so more before we're ready to start the development of that common language. We could rush it and we could mess up. I think we need, I'll pick a number, another year, another 18 months, another two years before we really understand this kind of system well enough to be able to start developing a common level. I'd like to see in the replies that people, both verbal and on the questionnaire, some sort of mandate for the program of the Systems Committee because it's really what this tenth anniversary meeting is all about. We're looking for direction, and we hope that you, the people that have been invited to this conference, will give it to us. Thank you very much indeed. [Applause]

MODERATOR:

Thanks, Ollie. Tax?

METAXIDES:

In spite of Bill's protestations, we do agree about some things that transpire. I really would have liked a couple of hours or more to discuss the work of the Database Task Group because I think much of the proceedings of the last two days have gone on as if no work was going on within the bounds of CODASYL. A number of us have been working pretty hard for two or more years; it's as if that didn't exist. However, in the time frame, I'm going to restrict myself to answering the questions and hope that some of the discussion afterwards will give me further opportunity. I'll read the questions first so that you know which question I'm answering.

Should COBOL be extended to handle databases more effectively? My answer is unequivocally yes. A language which is primarily aimed at data processing, and a heavily used language by business, simply must be oriented towards storing and retrieving data from a large database. Without such extensions, it simply won't cut the mustard. Two points arise out of the question itself: One, what do we mean by "extended"? What I mean is add language statements to permit interfacing with data management facilities. There is no implication that the facilities themselves must be included in COBOL. There seems to be quite a lot of misunderstanding about that. Number two is, what is meant by "handle databases more effectively"? And what I mean, again, is extend the capability of COBOL to store and retrieve data in secondary storage, based on a provision for declaring data structures, which simplifies system design, reduces the procedural coding content of programs, and increases the efficiency of execution. Note here the separation between procedure and declarations.

The Database Task Group, which is in process of developing a proposal for its standing COBOL, makes this distinction between what we're calling a data manipulation language and a data description language. The DBTG has some of its specific objectives—and I think it's important to know these to put the thing in context—to allow the declaration, creation, maintenance, and manipulation of an integrated database. To allow sharing of data without redundancy, whether the operational mode is online or batch, or both concurrently. To allow biasing of data to multiple application. That is, separating the logical and physical considerations, and providing for orienting data towards the specific processes you wish to perform. To provide and permit a variety of access methods. To relieve the programmer of physical placement considerations so that he can work at the logical level only, but to provide the data manager or the data administrator with control over physical placement because of efficiency considerations. To allow a variety of data structures to be declared from sequential to networks so that one can actually represent the data that you're work on in the way that you logically see it from the point of view of the processes you wish to perform instead of having to distort the data structure to what the software will permit. To relieve the programmer of maintaining such data relationships, which have been declared in that way.

The DBTG, in its first go-around, is not covering extensions for the nonprogrammer. However, we are paying particular attention to developing an approach which allows such growth. We simply can't do everything at once, and we're saying that the first thing is to develop and extend the language for the application programmer.

One last point on the first question. I think that any extension to handle databases is equally appropriate to other higher-level procedural languages such as FORTRAN, PL/I, ALGOL. Although we are doing our work in the context of COBOL, we have come, I think, to hold the opinion strongly that our work is equally applicable to other languages and not restricted to COBOL. I think I can bring out why that's so important; it already has been brought out several times.

Question Two. Should the extended system be able to operating on an existing COBOL data file? My answer to that is yes, but I'd like to elaborate a little bit. I don't think that the extended system should be constrained by what is allowable in existing COBOL data files. If in the 1850s a constraint on transportation development had been that existing canals be used, in which there was a huge investment, just as there is in data today, we wouldn't have had an industrial revolution at all in all probability. The development of the railways didn't stop large transports; it just changed the economics of the situation. It should then, in my opinion, continue to be possible to operate with existing COBOL data files and programs. But if the extensions are a real success, then you shouldn't want to for very long. The DBTG approach in no way interferes with the development or running of existing programs. However, to get the benefit of the extensions that we hope to propose, that will require some conversion effort, depending on the extent of the change that you wish to make.

Should the extensions make COBOL a tool for a database administrator? Yes. Systems programmer? Application programmer? Yes. Nonprogrammer? No, provisionally. COBOL is a procedural language at the moment. I know discussions went on that procedurality was a scale, and that is very clear. However, I think that we can all recognize the difference between COBOL and generalized systems of the ilk which appears in the Systems Committee output. COBOL, as I say, is a procedural language, and as such is more appropriate for handling the massive volumes frequently met within the day-to-day operations of many businesses. Since any business transaction is not an isolated event, but impinges on many other aspects of the business, extensions which facilitate more accurate modeling of a business make it even more appropriate that the language is useful for application programmers. So my answer for application programmers is definitely yes. Since the procedural language statements must exploit an interface with pre-declared structures, procedures and declarations being rather lack of cart and horse, they go together, but they should not be inseparable. That is, it should be possible for the declarative language to describe a database to exist on its own, and for any language to interface with that database by means of calls on data management software. Under these circumstances, whether you class COBOL as a tool for a database administrator is only a matter of classification. There would have to be a facility to describe databases. It should be freestanding, but COBOL, or any procedural language, must be able to interface with it. My answer to making COBOL a tool for a nonprogrammer was no, but let me quickly say that any such language must be able to interface with the same database as COBOL programs. However, such a language is nonprocedural by definition; it merely means perhaps that we need a new name other than COBOL. The point is I don't feel the question is quite right there. The important thing is that the levels of language developed must all be able to interface with the same data.

Should the extended system address itself to problems of database security and privacy? Yes, yes, yes. The words "extended system" I take to mean declaratives and procedural aspects of the language. In the social environment, as I call it, of an integrated database, the language required to interface with that database must include security and privacy features. Otherwise, you'll destroy the database. So somewhere or other, there must be privacy provisions. And security, I think Bill explained the difference there; I agree with him on the difference.

Should a new common language be developed to provide a database interface for the following: database administrator, nonprogrammers? Yes to both. I hope you will not construe my yes to the database administrator language as being a contradiction. Such a language should be an independent language; I've made that point, and I'll make it again. But it also must be a part of the procedural languages, which may be used to interact with a database. That is, they must be able to take advantage of the intelligence reflected in the description of the database. Read the nonprogrammer language? Let me stress once again I think any such language should be capable of interfacing the same database. As is developed as a result of the routine processing involving an integrated database interfaceable by N procedural languages. It seems to me much of the development today does not recognize this fact, and I think that is a crying shame. We have repetitive operations to perform. They are of huge volumes, and there simply isn't enough time in the world to not take a relatively biased approach to them. By that I mean biasing the data towards the solution of the problem. By definition, the nonprogrammer doesn't know what he wants ahead of time, and therefore there's no biasing of the data towards his needs.

If there is a need for a new common language, can the necessary simplicity for nonprogrammers be achieved? My answer to this question is it depends. For the kind of processing which does not demand what I call selection of selection criteria, I think the answer is yes. If, however, there is a considerable amount of analysis involved in the processing to be performed, then the nonprogrammer would, in fact, find himself writing a procedural step-by-step program, and we've gone full circle, and I don't see that that is very nonprocedural. If you have a complicated input which has to be analyzed, then there is some degree of procedurality implied in that.

The last question is, should such a new language be able to handle existing COBOL data files? All I'll say on this is that it must work with the data available, too, and this is clear repetition and updated by the bread-and-butter processing that goes on day in and day out. If you are using non-extended COBOL for that purpose, then it must be able to work on that data. Thank you. [Applause]

MODERATOR:

Thanks, Tax. Charlie?

BACHMAN:

Good morning. This topic we have today is one of particular interest to me. The aspect of the Database Task Group work, which Tax has just described with you, is one that I've been with for a considerable time. I think Warren Simmons, who is around here somewhere, and I and a few people met on a one-day project in November at least three and a half years ago in Phoenix, and that was kind of where the Database Task Group got kicked off. It was formalized somewhat later, meeting deliberately for that purpose in the spring of the following year, which would make it '66. But I'm not going to try to address myself specifically to the questions, as gone through here both by Bill Ollie and Tax at this moment, mainly because I very largely agree with what they've said. I think that between the two of them, there's more overlapping agreement than there is disagreement. I think that it has to do with whom you're trying to serve in the different markets. Bill Ollie sees the market consisting of many people who are not programmers as we now think of them. Or certainly, they're not professional programmers. If you look at some of the things that have happened in time-sharing, where language is like BASIC, which were extracted out of ALGOL in a sense, it really opened a new market of users. I think we're interested in opening that market of users for databases, too. Others—and I think Tax represented me this position—represents a large corporation with immense amounts of data to be processed. He's concerned more about feasibility, so he's got a problem of quite a different magnitude. Both of these are realistic situations. My view tends to follow the need for both of these things, and we'll see both develop.

I think in the beginning, I'd like to try to make one point with respect to database descriptions, database manipulation capabilities. I think it's a very important policy issue that CODASYL must think about and work with. This is this area of languages which are new extensions or a new language interfaced with COBOL. The problem is what do we do-- If we say this is part of COBOL, it helps us in one sense that the database language is part of COBOL. It happened to turn off another audience that we're also interested in. There's a certain set of people who feel comfortable in the FORTRAN language, as an example. Regardless of how difficult it may be to do some things in FORTRAN, they know FORTRAN well enough and they're comfortable with it, and they can solve fantastic data processing problems. FORTRAN is being done all the time. So we need to supply to those people who feel that FORTRAN is their native language. You'll need to supply those in the future who think PL/I is their native language. Capabilities to get the same database. Because corporations cannot allow the parochialism involved in having databases tied to a language. Somehow we must accomplish this across-language capability to support a database. We need to interface it with COBOL. For my own purpose, this is the most important first step, to get the COBOL programmer interfaced and operating effectively with this database.

We have an exactly equivalent situation in the message control area. That we cannot allow, in the same sense, COBOL message systems. They're going to have to be systems which go across not just sites in the same company, but also across sites in different companies, different machine lines. So we have a whole set of areas, and let me just enumerate them for the moment. We've got the database management area. We've got the message control area. I think we have job control management, in the same sense, across machine lines. We need to discuss those, not on a per-manufacturer basis, but on a user, "What's in my problem and how do I get at it?" basis. Probably, to a lesser extent, we've got the programmed control language. How do I build, manipulate, and operate on programs? Not when I'm trying to execute them, but when I'm trying to build them and manipulate them and test them out.

Now to deviate from that and come back specifically in the database management area, I'd like to describe, in one sense, some words we've been using in Phoenix to help understand the difference between what Bill Ollie's driving for and what Tax is driving for. Bill must have a mental model, a conceptual model. We're talking about being small, medium, and large file systems. What constitutes a small system is a very arbitrary kind of a definition, but "small" is relevant to certain factors. Now, a file is small if I look at, say-- well, the machine size is one thing to consider. A given file, let's say, of a million characters on a very small machine has the attribute of being large. That same file on a great, big machine has the relative attribute of being small so that the factor of machine size—how much core is available, how fast is it—certainly affects the strategy in which you go about driving that file. In the same sense, the file complexity. We're talking about simple unit records that can be thought of being sequentially or randomly accessed. If a file complexity is low, the file conceptually tends to be small. File activities—another aspect. When a file gets beat to death—and someone mentioned the other day that they had implemented here, they implemented a message control system written in COBOL. It was handling something like 180,000 line items a day. In that sense of activity, that's a large system. In the same sense, we talk about people. The persons build the tree of the file. If a person himself is not a professional programmer, then his skills of treating this file are somewhat less than those who are professional programmers, and this will affect the size of the file.

I think we must be prepared, in many ways, to provide systems that support small files, systems to support medium files and large files. The Database Task Group, at the moment, I think, is largely motivated to support the problems that tend to fall on the large file area, because the corporations that have been meeting with us have had files that are beyond the magnitude of even the hardware you can find today. Beyond the magnitude of the software that can support it. So in this sense, these are large files by definition.

They're stretching our capability to handle them. I think five years from now, we'll have better hardware and better software, but there'll still be large files because man's capability and desire to increase the scope and the command of data via systems will always exceed our capability to handle it. I don't think there's any reason to not think that's true. So we'll always have a large file by that definition. And always be stretching for systems which allow some hand-tailoring, some engineering, if you like, to that file design.

One of the things that we've been working with is a procedural language, these languages being interfaced with COBOL. It's our feeling though, and I think this echoes, in some sense, something that Tax Metaxides said a few minutes ago, that given that large file, given the capability to handle it in a procedural sense, to do the kind of things which are hard to do in a nonprocedural way, that you also need things like report generators to extract reports for it. Well, in a sense, a report generator is a very nice case of a nonprocedural language, but you tend to talk about, "I want to describe my report." I describe what the file looks like, and I describe certain selection criteria," and essentially say, "Go to it. Get my report for me." I think in another sense, we've talked about inquiry systems. An inquiry system typically has even less complex statement of the problem. It wants all the records for which certain things are true and wants to print certain fields from it. So this, again, is even more nonprocedural language.

I think that, in some sense, we forget that many of us are already operating in systems today with exceedingly nonprocedural language statements. If you go into most businesses and say, "Well, how does someone handle an order in a computer?" They say, "I type out the order on some form, and they're on punched card or paper tape." You, in effect, have one command, so you send this to a certain destination. That's the only, really, command on that. That's a nonprocedural statement. Now, it happens to be a programmer associated with that destination. That's been fully planned, to treat an order processing. So we're saying, "Process that order." We have that data; it's that ultimate sense of nonprocedural. That's much like the joke about the prisoners who use to call out numbers and laugh at the jokes because they knew them so well. I think we must recognize we have that capability today to hit the nonprocedural; just call it by number.

One of the areas that (and I'm jumping from topic to topic just to kind of fill in some areas here) there's been relatively little attention paid to, and this is the area of shared database processing. By "shared database", I mean the capability of having more than one program, more than one run unit, in a COBOL concept concurrently operating against the same file, and potentially even the same records. An ability to do this thing efficiently, to do it under complete control to maintain the integrity we've talked about, and still have some way to let the programmer know what's going on. In the best of all ideal words, we'd avoid this, but it's quite clear that our available forms of mass storage—be it drums or disks or magnetic strips or newer electronic types of things—are all still going to give us more capability if we attack in parallel. It's over a long time, and probably forever, we've been trying to use parallelism as a way to get cleaner throughput. Parallelism is a thing which generates the shared access.

One thing that the Database Task Group looked at very closely is embedded in the current status report. Just an example: This is the size of the current documents. There's a lot of work being done. Now, this is only printed one size to try to help you a little bit, but there's a considerable amount of work that's been done, as Tax had described it. It's not quite at the point yet that we feel free to say it's acceptable individually or collectively, but there's plenty of work that has been going on. So in this language, we've been concerned about, how do I handle the aspect of shared access? How do I let two programs get the same data record, essentially simultaneously? Under the right circumstance, we'd be freely happy about it. On the other hand, if both of them tried to update this inventory record or seats on the airline reservation system, the number of seats available, you catch what happens in case this goes on. We've got to have these facilities, and these are the kinds of things we're treating in these documents in terms of maintaining the integrity of the file. In another sense of integrity, we're looking at the capability to really interject some things you might call job control elements. These are elements like saying, "I'm going to take a checkpoint." Not just of the program, but the database. Because in case something happens along the way, I'd like to be able to roll back the user database to that point or the program to that point and maintain, again, programmer control of what goes on. So these kinds of things are being considered, and being considered very carefully.

In the area of privacy, we've talked about the ability to initiate privacy at many different levels. Privacy, obviously, at the file level. But more specifically, privacy at the record level, and even privacy at the data item level so that you may be able to get access to certain payroll records, and quite commonly in terms of inquiry systems. Inquiry system access to this file. And to get a person's name and address and things of this type. His home phone number or his office phone number. That information about his salary or the date on the last review would be quite information as to where it is and who it's available to. So all these things are being attacked, I think, very carefully. One of the things I hope to come out of this meeting, through the discussions I've had personally and through this meeting today, is to find out what kind of support we have to push along in this direction. So I invite all your comments in this direction during the discussion period. Thank you. [Applause]

MODERATOR:

Thank you, Charlie. We will now hear from Dick Kerrs.

KERRS:

I can't question the need, certainly, for work in this area, and the fact that it can properly be done within CODASYL. I might say that--

[CODASYL Disc 7]

KERRS:

--on the subject are cautious, and I could say that they stem from my respect for the problem, or from my recognition, at least, of the complexity of the proposed solutions to the problem. Or even my realization of the expense involved in the implementation of any one of the proposed solutions in the hardware environments of today or that we think we're going to have tomorrow. If, however, I look deeply into my heart, I find my thoughts are truly driven more, I think, by stark terror at the thought that I might have to implement and use the thing. The thought of what we could do, in effect, to my friend, COBOL. He's built a house and raises four kids, so I have to take care of him. My thoughts are probably oriented toward what it could do to COBOL, and I don't mean that we shouldn't do anything, just that we should be careful. I'll scroll through the questions, as everyone else has. Give you my impressions. And again, they might be a little bit aimed that the Programming Language Committee and how we relate to them.

What do I think about "Should COBOL be extended to handle these databases more effectively?" Just effectively. If these extensions we're talking about are consonant with the aim of COBOL, yes, I don't think we should try to extend COBOL to solve all of the problems of the database. I think I have some agreement there. Should this extended system be able to operate on an existing COBOL data file? If the extensions are consonant with COBOL, I think they must. Should these extensions make COBOL a tool for the database administrator systems programmer, application programmer, and nonprogrammer? I have the feeling that regardless of what our intent was to aim COBOL, it is aimed at the application programmer. I realize that we want to solve the systems programmers' problem. I think COBOL was meant to be a step in that direction. We need to solve the problem for the data administrator, for the systems programmer, and for the nonprogrammer, but not particularly through COBOL, I think. Should the extended system address itself to problems with database security and privacy? Well, I don't see how we could implement it without addressing those problems and solving them. How we'll do it, where we'll get the manpower to do it, and the money, I don't know. Perhaps CODASYL could arrange to have, with each integrated file, delivered a fuzzy blanket to the administrator. "Please take care of the security side." [Laughter]

Should a new common language be developed to provide database interface for the following? Well, if I'm to be consistent, I would say just not COBOL. If "a new common language" means not COBOL, I think so. We can design something for the database administrator and successfully, as 5.6 asks, for the nonprogrammers if we do our homework and satisfy the needs of the people in the middle—the application programmer and the systems programmer. Once the system is slick, I think the problem of interfacing with the nonprogrammer should not be too difficult. Should such a new language be able to handle existing COBOL data files? If the new language is aimed at the nonprogrammer, and I don't believe that COBOL programmers of today are nonprogrammers, or they're not nonprogrammers. It probably would be nice, but I don't know that it would be so important to interface the people who are now nonprogrammers with existing files of today. My concerns, of course, as I mentioned before, are aimed at this relation between the database management and COBOL. With the thoughts that we covered earlier this morning about the data descriptive language, if these things are brought to fruition, please don't burden the COBOL programmer with describing and redescribing the data structure that he interfaces with, if you agree with me that he's an applications programmer. And above all, don't try to spread COBOL across all of the users that we've mentioned here—the administrator, the systems programmer, the application programmer, and the nonprogrammer. [Applause]

MODERATOR:

Thanks, Dick. Chuck?

GREENBERGER:

Well, you can tell Mel and I are from the same company because I'm going to start out talking about objectives or user needs and problems, because these are very closely related. I think the success of COBOL historically was due to the fact that it met some well-defined objectives. As I recall then, they would have cut the cost of program development and maintenance by providing a higher-level language. I remember charts that Jack Jones presented about the costs of government programming and maintenance of programs, and the associated documentation, and how we're going to get savings. We also had the objective of getting a language which would be, in some sense, usable across manufacturers' machine lines. These objectives, as I raised the question earlier, have to some extent met.

I think it's important now to talk specifically about what the user needs, ergo objectives, maybe in the years 1972, '73, and thereabouts, as we can best forecast them because that is a time span needed for implementation of something new. Now, I'll discuss the needs as I and a representative of a group of people in my company see them. Many of these points have been covered in various points of the conversation over the last day and a half, so bear with me; I think I'll end up with a different point of view. First, we agree that many programs will interact with one collection of data—numbers and letters. That's been said many times. There are reasons for this. We will have files, which will be used across classical application lines. Secondly, we will need to distribute work among people. Different people will do different work on the files, both in development and the use of the files. Secondly, programs will be written in several languages—COBOL, FORTRAN. Now, in our decision-making systems, we use accounting data to bring in costs and to an LP matrix. That's one example. We also will be using, à la John Young, terminal languages to inquiry on files, databases, that have been set up in maybe an accounting run. So we see this kind of need coming along. I can't claim that we have a lot of applications that were actually in the shape. For one thing, it's not feasible today. We don't have what I will call a data management system, so we don't set up our systems this way. But it's a need and a pinch we're feeling.

Thirdly, our collections, files, bases are dynamic. They will be changing in structure and logical structure—you know, the connections between things—and also the kinds of things that are in it—generally, this will expand. We'll keep adding new stuff about people, say, to our personnel files. We'll find new things we have to add. So we have to have some facilities to account for dynamic changes in files. I have certain implications about how we structure our approaches here. We also, as in the past, need reasonable efficiency, both in terms of response time, hardware cost, and people time to use this stuff and create the stuff. People time and people accessibility becoming more important than hardware costs. Also, in our company, we have different locations with different-sized equipments. Some very small, little 30s, model 30s (that's an IBM machine), and some big machines. These have different needs. I mean, they just don't have the capability of doing some fancy stuff like online systems. But we would like to have, for training purposes and in people interchange and data interchange, consistent approaches across these different sizes and with these different needs.

These things kind of add up to some stuff, which I will call a data manager. To make that just generally understood, a data manager's a systems software, in my terminology here, which provides data to different applications programs and also does a lot of housekeeping on the data files, so it's used by the database administrator. I call that a system because it does some different things; it may not be one thing. But I'm going to use the word "data manager" for this kind of software stuff. The kind of attributes we'd like to see filing from these needs is that the data manager should be separate from the procedural programs. That's been covered. It should be callable by all procedural languages. Not necessarily procedural, but nonprocedural. We are, in most of our big systems, developing an applications-oriented language, so that the guy has, in effect, on our payroll application or personnel records application, an inquiry language for his own application. He learns that, about how to call on people. As John has correctly pointed out, this tends to be more nonprocedural on direction. It tends to be less flexible because it has a syntax that the people who work with human records understand. We tend to build this kind of language for each of our major application areas. This is arguable; some people may say you ought to use it to completely generalize language, but this is what we're doing, and this is what we see more of coming.

We also see that the data manager should provide a wide range of capabilities. This is the result of our differing needs. We have some places where we need simple stuff, like sequential files with sequential access. But we still need to maybe get at that file with FORTRAN and COBOL and some other stuff. We also have some more complex needs. We can see hierarchical structures, different logical trees, what have you. The whole smear of interesting geometries among files. We can see a hierarchy, and of course, we can see that we have different needs for security. We have some stuff that we would just want to lock the computer room door on. We also have different needs; we may want some program security when we have terminals scattered around in unlocked places. So we have a wide variety of needs, and this really call for a modular approach to things if they're going to be consistent. We also feel that a data manager should be open-ended. We don't think we know enough now—and this point has been made before—of how to use our present gear. We just don't have enough user experience on this kind of stuff. We have an anticipation of new hardware coming along that might require different stuff closer down to the physical machine, and might even change our ideas about logic and organization. So we'd like to have some open-endedness here. Lastly, we'd like to interface this between the data manager and the operating system, which is a big hunk of software, and the procedural languages, and the communications software, and other software things but these are the big things in our mind, to be clean. When you talk about security, you know what the operating system is responsible for in that dimension, and the data manager is, and what the procedures are responsible for. You don't get into kind of the thing that we get into in COBOL now, cleaning up our divisions. We like to make sure what our divisions are early on. So those are kind of the attributes I sorted out.

Now, I talked about data management independently of CODASYL. I'm not going to get to CODASYL and what I think CODASYL ought to do. First of all, I see some problems for CODASYL in meeting these requirements. One is the dedication to COBOL. Tax has said the first order of business is to extend COBOL and to talk about language, and I just don't agree. I see traces of COBOL thinking throughout the organization, and these are very deep. The second thing relates to this, is the community's image of CODASYL. They think we're a COBOL committee, so it may impact on the acceptability of any CODASYL developments. Gloom and doom. That's a problem. Secondly, I think we need full-time people on this of highly qualified professionals. This is a problem within the CODASYL framework. There can be ways to work it out, but I think on problems of these magnitudes, part-time efforts are ludicrous. And the monies that are going to be expended both using it and implementing it. There have been precedents for full-time COBOL efforts. We need to find a Short-Range Committee. We're able to get people broken away from their companies for three months to do a three-month job.

HOLLIS:

Not true.

GREENBERGER:

It wasn't true. Okay, sorry. I thought it was pretty full-time.

HOLLIS:

Only three weeks out of the month away.

GREENBERGER:

And they first want to send an audience debater. [Laughs] It's not an important point. I just don't think part-time efforts are desirable, in my opinion. Another area is the probability of implementation of the recommendations. I know this has hung up some of the efforts. People feel, "Well, we can specify some nice stuff, but what is our force of implementation?" One point of view is quality will be the force of implementation. Just our prestige, the quality of our people, and the quality of our product. Well, as Howard pointed out--

[CODASYL Disc 8]

MODERATOR:

Jack Jones has asked me to announce now, so that the questions don't plague you all morning, that the roster of attendees will be sent out to you shortly. It will not be available here today. It will be sent out. Along with it, we hope we will be able to synthesize the observations that are provided in the questionnaires, if you'll please turn them in today. And any general observations which the emperor's exalted group, the Executive Committee, will make in their meeting tomorrow to join both things together.

When I said good morning, I meant that is probably the last accurate thing I'll say in this session. It only is accurate insofar as the Washington environment is concerned. As far as physical environment, I'd have to be an optimist to predict on how anybody felt this morning, but it's a pretty good turnout. On the other hand, most of you know what an optimist is, I guess. He's a guy who goes down to the Marriage License Bureau once a year to check and see if his is expired. My wife doesn't like that one. That's bragging, isn't it? You know, there's an observation that I made yesterday. I'm kind of surprised about two things. Having been old enough to be old a the first session ten years ago, and seeing many of the folks who were there here, ten years later, there are two things surprising: how few of them have gotten any older, and how affluent most of them have gotten. You'll notice I said that in the third person.

To start this session off—I hate like hell to steal time from Jean Sammet, but Bill Ollie just reminded me that we only have a half-hour. We're going to talk about the subject of procedural and nonprocedural languages. I think we're going to blend the schedule a little today. If we run over five minutes, we'll try and adjust. We want to keep the comments and the repartee going. Without any further razzle-dazzle from me, I'd like to turn the session over now to discuss her impressions of procedural and nonprocedural languages to that distinguished authoress, Jean Sammet.

SAMMET:

He's much too hard an act to follow, and I guess my only comment, I can't compete with his jokes, but the one thing that I must admit that amused in the process of quite a bit of very hard work while working on a book was the fact that I was literally working every weekend and on nights. The building that we had has a guard on Saturday and Sunday, and after seeing me come in every Saturday and every Sunday for a number of months, the guard turned to me and said, "Why are you coming in so often?" I said, "I'm trying to finish a book." He said, "My, you must be a slow reader." [Laughter]

The term "nonprocedural languages" is one that I personally find rather obnoxious at this point in time because it's been bandied around with a great deal of hot air and very little light shed on the subject up until now. Frankly, I tried for quite a while to define what was meant by a nonprocedural language. I didn't succeed, and I thought that was some failing on my part, and I went around looking for other definitions, none of which seemed to be very satisfactory. I finally reached the conclusion that whatever other failings I might have, a failure to define this was not really one of them because it is really a relative term. This is the main point that I want to get across as at least my opinion; namely that the definition of a nonprocedural language is something which changes as the state of the art changes. That's true of lots of things, as somebody pointed out to me yesterday. But I think it's much truer here than in many other cases.

For example, we may have something that requires a certain number of sequential steps, which can then be grouped into a larger unit in which you don't actually have to specify these; in which perhaps the system can determine what it is that is to be done. But let me give you a very simple illustration. Back in the, some people maybe should pardon the expression the "A2 days" on similar systems of that kind, when you had maybe a three-address pseudocode or something like this for a person to be able to write the statement y = ax + b, and assume that if he had given the values for a, x, and b, that this would indeed be calculated by the system, if anybody had thought in those terms, would have been considered a very nonprocedural statement because there was, in fact, no system-- I'm saying prior to the time that there was any system that could accept the sentence that said y = ax + b. That would have been considered a statement that had a lot of sequential steps in there that the user had to state for the system, and the system could then go ahead and perform the computation. Of course, since the advent of FORTRAN and similar systems, obviously we have long gone beyond that, and we no longer have to worry about y = ax + b as being an acceptable sentence or an acceptable statement to a system.

However, let's considering the following the following two examples, and unfortunately, my favorite and best one is a little more mathematically oriented, but I'll hope you'll bear with me. Suppose I say, "Calculate the square root of the prime numbers from one to 91, and print in three columns." Now, there's no system right now that can accept that statement. If I put this in, it's just going to blow up in any system that I'm aware of, either in existence or even on the drawing boards. Similarly, if I take the statement that says, "Print all salary checks for nonexempt employees," the system will also blow up because there is no system capable of accepting that breadth of generality and that amount of—if you'll again pardon the expression here—nonproceduralness. These are not specific terms; they change. And if, in fact, we were to design a system, and in fact I designed a language which designed but never implemented a language, which indeed could accept the statement, "Calculate the square root of the prime numbers from one to 91, and print in three columns," the moment that that becomes an acceptable statement to a system, we stop thinking of it as being nonprocedural. Similarly, if I want to make any kind of statement about "Print salary checks," or "Find the number of the people who do this, that, and the other thing," again, the question of whether that is nonprocedural depends very much on the state of the art and not on some absolute, fixed relationship.

The kind of thing that I think is needed to shed some light on this nonprocedural question is some kind of numerical scale, which can be established once and for all, and whereby we can measure statements and whereby we can measure systems. That is to say, we would have a scale maybe running from zero to one with all the real numbers in between, and we could say that a statement or a system or a language was at such and such a point on the scale. I think that could be done and done in a way which provided some fixed points of reference. And then as the state of the art of the compiling art, of the translating art…as that art improved, we would simply be able to say, "Well, now the state of the art allows us to accept statements which have .79 nonproceduralness or .2 nonproceduralness," or something of that kind.

There have been several attempts, much more in the mathematical area, to improve the state of nonproceduralness, and I won't bother you with those references. The areas in the more data processing-oriented applications certainly do exist. The best example of this, as far as I'm concerned, is the report generator, in which, indeed, you specify to the system what it is that you want done, but not exactly how you want it done. But even within report generators, there are still elements that you have to tell the system how to do it, as well as what it is that you want done. Again, we do not have a numeric scale for indicating how far along this nonproceduralness route we have come.

Stay with this topic, but shift over a subject which brought this home to a number of us in a very strong way. I'm surprised, and I guess a little bit disappointed, that in all of the history discussion and what have you yesterday, there was only one very brief mention made of one of the fairly active and productive committees under CODASYL and, with all due respect, to the reference that had the wrong name for the committee, namely, the Language Structure Group. We did produce a paper called, "The Information Algebra." Now, the reason that I raise is that one of the objectives of "The Information Algebra" was to provide a very broad systems analysis, nonprocedural-type language for data processing. That paper was, as correctly stated, published in April of 1962 in the *ACM Communications*. I might also point out that we voluntarily voted to disband. It is probably the only committee in history that has ever voted itself out of existence. The reason that we did it is because we found that taking the next steps within that framework were just too difficult. It would be the kind of thing that would really require a full-time effort and full-time participation, and none of us could provide that. Or more importantly, none of our employers could or would provide that. But there we had a kind of situation which literally, in one line of "procedure", one line of instructions to the computer, could be stated how you would computer payroll, and then there were a few steps of data stated somewhat separately.

Now, if you have one line telling a computer—and it was a fairly abstract mathematical line in notation, at least—if you have one line to a computer telling how a payroll should be computered, by any reasonable standard today, that'd be pretty high on a proceduralness scale. But what we found when we started to look at this was that within subunits of this, there were indeed procedural steps that had to be followed. No matter what it is that you tell a computer or tell a system to do, you will, in fact, be constrained to follow some things in sequence. If you go back to my earlier statement, which said, "Calculate the square root of the prime numbers and print," it's perfectly clear you carry out the printing after you do some of the calculation and not before. It's a question of who's going to determine what sequence is to be applied. So we found that even where we had something that was really considered quite nonprocedural in any scale of the kind I'm talking about, although we didn't use those terms, nevertheless, it did have many elements of proceduralness within it.

Another example that I want to give you of, in some sense, nonproceduralness systems (and you'll notice I avoid the word "language") is a sort generator, which, again, you tell the sort generator the kinds of things that you want done, but you don't tell it how to go about doing it. Now again, if you consider COBOL prior to the existence of a "sort" verb, you could say that to carry out a sort computation was fairly nonprocedural because the system couldn't accept the statement that said, "Sort." Well, indeed it can now. As we go on further and further, we will be able to deal with more of these systems that will accept broader and broader statements.

Now, the questions that I'm supposed to address were the two, of which the first one I have to disclaim as being somewhat meaningless in my terminology, although I think it is a legitimate question from the point of view of people who phrased it. The gist of it, I think, was, "Will nonprocedural languages head towards replacing procedural ones?" In my terminology, my answer is only that the question is meaningless; we will proceed further along the scale of nonproceduralness, but that the question itself can't be answered. That gets me off the hook on the second one, which says, "When will this be done from a date point of view?" And since I content that the first question is meaningless, I can't quite give a date to when it'll be accomplished. Thank you. [Applause]

MODERATOR:

I'm not going to do it, but I know when I'm going to. No, it's the other around. Our other speaker this morning now is Mr. John Young of NCR and Hawthorne, California. He brought some of the California weather here with him, I think.

YOUNG:

Since Jean has already been involved in one exposé during this meeting during the course of the lunch yesterday, I want to make it clear that we did not spend the night preparing a common approach to this problem. The coincidence that you will note between our approaches is exact that, a coincidence, because I'm a last-minute replacement for Duane Hedges, who was unable to be here this morning, and so Jean and I haven't even talked about what either of us was going to say.

First, as Jean did, we should take a little harder look at the definition and context of nonproceduralty. I hate arguing about definitions. I have a little slogan, which I developed initially in connection with the term "time-sharing"; I think it applies equally well to "nonprocedural." As originally stated, it said if we had spent as much time on time-sharing as we have on the definition of time-sharing, we'd have time-sharing.

I think it was discussions a number of years ago, particularly led by Mike Motobono [?], which led to the statement, "One man's solution is another man's problem." This is the point that Jean made, that there is a hierarchy of these problems along the course of any project. Let me take an enterprise; for example, the requirement, "Build a bridge." This is a nonprocedural statement. However, this solution to the problem of bridge building is somebody else's problem, namely the person who has to sit down and write out the statements, "Design span," "Design peers," and so on down. "Buy material." You know. "Sink concrete," and so on. Let's take one of these procedural statements now out of this sequence, like "Design span." We're now going to break this down again into a procedural sequence—design, support, calculate weight, calculate bending stress, calculate strength, compare the two latter elements, and if the strength is less than the stress, the bridge will collapse of its own weight and we branch back to the beginning of this little loop. Now let's assume that we go through this procedure a number of times, and we come down to a particular calculation which we need, "Calculate weight." We can put this in a nonprocedural form by saying, "Calculate weight (2,500 feet, 300 cars per hour, 60 miles per an hour wind)." This now will calculate the weight of a bridge span which will meet those requirements.

Okay, again, we have a nonprocedural statement which we can now break down into a number of procedural steps. Let's suppose that one of these procedural steps is the solution of a quadratic equation. We can approach this in a number of ways. For example—and let's assume for the moment, since that's my hobby, that I'm working at a display terminal. I can call up a form, which is headed "Quadratic Equations Solver," and which says, "a," "b," "c," and then "x = [blank], [blank]." I fill in a, b, and c, I push the button, and now the two values of x come back. Now, the next kind of input that I can make is to say, "Solve 253x + 17.9 = 0," and send this back, and I will be presented by the system with the two values of x. I could go to a FORTRAN statement. I could say, "x = (253 + (…" and so on, go through that. And finally, if I want to be a real purist, I could go to the form, "Load b. Multiply by b. Store in p. Load –4. Multiply," etc.

Now, this is where Jean's and my presentation really do come close together. I think you can replace the idea of nonprocedurality with a quantitative measure. One measure that's been used in this area is based on the concept of an atomic symbol, which is a name, an operations sign, a number, and so on. If you'll look back to the example that I just gave, the fill-in form for the quadratic equation solver had three atoms where it required the user to state that much information. The "Solve x^2 + etc." had 12, the FORTRAN statement had 17, and the assembly-level language had 23. Now, if we wanted a way more complex problem, for example, a cubic equation, the solve statement, or the input form, goes up very slightly. Solve goes from 12 to 14 atoms. But if you think for a moment about how you solve a cubic equation, you'll see that both the FORTRAN statements and the assembly level go up much more sharply. The situation stabilizes around "n = 5" because there you're going to have to go to some kind of effort to procedural, which will be a fairly constant amount of statement, regardless of the degree of the equation. I think we see here then that we could replace nonprocedural by a quantitative measure called something like, "Degree of noncomplexity." We saw in the example that we moved from a noncomplexity of three atoms to something like 23 for assembly-level language.

Incidentally, another important point, too, I think, is the fact that implicitly, or psychology at least, people view nonprocedurality as being a binary variable. Languages either are or aren't nonprocedural. I agree with Jean that this is a sort of meaningless idea, and I think that a continuous measure is a much more sensible one.

Okay, now we have a beautiful, noncomplex solution to the quadratic equation problem. We might think back for a moment at that sequence of techniques and note that as we got more complex, we also got more flexible. Our quadratic equation solver would do just that—solve quadratic equations. Our "solve" verb could, in theory at least, solve polynomials of any degree, and finally, we get down to an assembly language that can do anything at all, albeit not very well sometimes. Okay, now my next equation is something like $x + 17^{3x}$ – [hyperbolic sign]$x^{3.5} = 15x$, and I'm right back where I started. Now I have to go back to some more flexible, more complex language to solve this new kind of equation.

I think the major point here is that noncomplex languages-- I might point out that I think we're going to have to change our thinking a little bit to include as languages sets of forms, graphic display inputs, and things of this kind. Instead of saying, "Design span," I might very well draw in a picture of it on a graphic console. At any rate, these noncomplex languages give us a tremendous advantage—as all of us, I think, clearly see—so long as we stay in a domain of known problems. So long as I was working with quadratic equations, I was fine. As soon as I moved to transcendental functions, then I was again in trouble. To the extent that the system designer knows or can guess the requirements of the user, he can provide to the user a noncomplex language to solve his problems. Unfortunately, there's always some kook in the organization who thinks that he knows how to run his business better than the computer systems designer does. So we're going to find that the noncomplex systems will solve most of the users all of the time, and all of the users most of the time. As to when this happy event occurs, someone at a recent meeting said that we in the computing community constantly overestimate what we can do in one year and underestimate what we can do in 10. Thank you. [Applause]

MODERATOR:

Thank you, John. Do we have any comments or discussion? Questions? Well, the shared brilliance to the solution has blinded everybody. Oh, no. Go ahead. [Laughs]

McGEE:

Bill McGee, IBM. John, it's not clear, when you use the term "noncomplex" whether you mean synonymous with procedural or nonprocedural. I wonder if you could clarify that.

YOUNG:

No, I think what I really meant to imply was that complexity was a variable which runs continuously between the zero of nonprocedural and the one of procedural, or vice versa. It's something that covers the range of what has been attempted to be described by the binary variable nonprocedural.

McGEE:

But which end is which? Complex is procedural?

YOUNG:

"Complex" is nonprocedural. I'm sorry, "noncomplex" is nonprocedural.

HOLLIS:

The point that I recently made was that if you're going to go it nonprocedural where you're trying to make the capability of the computing power available to more people so that they can express what they want more easily, and I like to somewhat make some point, as I do with an electric light switch. We have many electric light switches that can do many things for us. But if we have it so we can dim the lights and put out some of them and so forth and so on, then this makes it easier for the user, but behind that switch, there is a great development. I think that this points need to be made, that the reason that we can go more the nonprocedural route is that we're building more capability into the system behind it. Therefore, this statement determines the rate at which we go in the nonprocedural way.

MODERATOR:

That was Mary Hollis. Yes?

FOGAL:

Dan Fogal with Chrysler. It seems to me that perhaps part of what they have said is that proceduralness, like beauty, is in the eye of the beholder to some extent. A light switch, in a sense, is nonprocedural to some people and it's procedural to others. An electrician would certainly think of it in procedural terms because he understands and assumes the procedure behind it. So the point that's being made about a FORTRAN statement, we think of as being procedural because we know the procedure that's behind it. If somebody doesn't at all, it's perhaps nonprocedural in another sense. So I just simply the continuum idea, but perhaps a lot of these things are nonprocedural only to someone who proposes to use it without understanding what the procedure is as well.

MODERATOR:

Jean has an observation.

SAMMET:

While I would agree in general with that, I think there is one slightly misleading element here in terms of what you said, whereas I would much more strongly support the statement that Mary Hollis made; namely that there is an absolute fact of whether or not a system exists which can accept this statement. Now, whether I as an individual, may or may not understand how the scan of a FORTRAN statement or the scan of an equation is made, or whether I do or don't as an individual, understand how a sort generator or a report generator or a database management system or anything like that operates, really is not very important. The key thing is whether or not the system can do this, and whether I understand how the system does it, is, as far as I'm concerned, completely irrelevant.

MODERATOR:

When you're playing around with our money, I bet the editors are going to insist that somebody on the staff what the procedure is behind the nonprocedurality.

BUSHED:

Anheim Bushed from MIT. I think all of those things are procedural or nonprocedural business [inaudible] at MIT. What if someone said at the nonprocedural level what you can just say, “Do all work,” and the computer goes and does everything that you want it to? I think what is sort of required is a model that you can go down, down deeper into it. Then you can go into the FORTRAN program, all the assembly languages. But it’s up to the system level to see what the system can do. Like the people can define their own support routines, their own functions, and can say, “Okay, solve the equation,” and the function exists in the system. Then the system will then solve it; otherwise you have to write your own procedure for doing it. I’m not sure whether this unit belongs in the definition of the language, because I don’t think one can anticipate the problems the people who want to do it, so you can’t rely the functions. So maybe what’s required is to give a good language so that the user can run this procedure, and once he has them, once they are in the system, he can use them. Therefore, once more and more procedures for doing the things that we would want to do out of the system, then the user who doesn’t know how to write the procedure could use those functions and could solve his problems.

MODERATOR:

We have time for one more. Howard?

BROMBERG:

This is a question. Do you think that nonprocedural languages will place severe constraints upon the scope and the flexibility of data as can be described today?

SAMMET:

No.

YOUNG:

It better not.

MODERATOR:

Those are rather quick and direct. Do you want to elaborate, Jean? You know, it’s kind of hard to get an answer that just says, “No.”

SAMMET:

It’s very easy. I can tell--

MODERATOR:

For me? Go ahead. [Laughter] I guess it's appropriate now. We're about two or three minutes over. I don't know why we only had a half-hour here, but we'll turn the session over now to the next chairman, Bill Ollie, and his panel on data description languages. Thanks, John, and thank you, Jean. [Applause]

OLLIE:

Good morning, ladies and gentlemen. This, at long last, is the session on data description languages. I say "at long last" because yesterday afternoon, the topic of data description languages came up so often that I got the feeling people were rather impatience and felt this was an extremely important topic and wanted to get onto it. Data description languages is, let's say, the latest buzzword in the industry. It's another of those things that everybody wants, and nobody really knows what it is. On the Systems Committee, we have not really given a lot of attention to data description languages. In the survey that we've just completed, we felt very keenly a need for a level of data description language. We did feel that in the admittedly very brief look that we gave the problem, that there are levels of data description languages. We felt, in this particular session that a flavor of tutorials was called for. It would be very easy to have a session where we just sort of say, "Okay, we're going to talk about data description languages. Everybody jump up and sound off." That is probably why the session outline for this session is longer than it is for any other session. I don't know how many of you have read it. I think it'll be very helpful if you have.

What I'm going to do, I'm going to ask Bill McGee, who is one of my colleagues on the Systems Committee. Bill McGee is with a company called IBM. Some of you may have heard of it. Bill is going to go through the questions and probably try to explain a little bit of the thinking that goes into posing the question, and also give his own personal answers to what he thinks these questions are. After Bill has spoken, we will over to John Gosden from the Mitre Corporation. John is Chairman of the USASI Ad Hoc Committee on Data Description Languages, the work of which was mentioned by Bob Bemer yesterday morning, and also by Charlie Phillips yesterday afternoon. I'm saying this for John's benefit because he wasn't here yesterday, so he has a feeling for the fact that this topic has already reared its head yesterday, and just probably avoid covering any ground that may already have been covered. So I'll ask Bill to take over.

McGEE:

I bring you greetings from Watsonia. The subject of data description follows rather naturally a discussion of procedural and nonprocedural in the sense that it's a subject which is common, really, to either approach to languages as it seems to come up in connection both procedural and nonprocedural languages. Data description is not, certainly, a new subject to people in the COBOL fraternity, if I can use that term. The data division of a COBOL program is data description exemplified. However, it's surprising, in talking to people around the industry, how many have never heard of data description, and who wonder, when you try to define what it is, why in the world you need data description. There are languages—and I'm sure you'll aware of these—which don't use data description in the sense that COBOL does. But rather the system, or whatever it is that processes the programs, will attempt to infer from other language statements the information that is normally conveyed through an explicit data description. So it's a reasonable question, I think, whenever we talk about data description, to always try to keep in mind the reasons why we try to make the descriptions, what are data descriptions useful for?

I think in the context of COBOL, there are really two basic reasons why we like to make data descriptions. One of these is that it permits programs to be written in a symbolic way. That is, we can use symbols for variables rather than using absolute addresses. Of course, this is the whole basis for the so-called higher-order languages. With it goes the consequent simplicity and ease of writing programs, and perhaps more importantly these days, the independence of the programmer from the manner in which data are actually represented. This is becoming increasingly important in the database environment, where, for various reasons, the database manager will want to change the method of representation from time to time, but he will want to do this in such a way that it doesn't impact or doesn't affect the programs that the application programmer writes.

The second basic reason why data description is useful is that it affords a certain degree of independence between the procedure, which the programmer writes, and the data on which that procedure operates. Ideally, one would like to be able to write procedures in such a way that they will work on a fairly wide class of data, so that one could, for example, change the data description without affecting the procedure. This is a goal that is not always realized, but at least this is one of the reasons behind an explicit data description.

In our outline here, we have attempted to identify four levels at which a data description may be made. The first level is a conceptual description. This is a type of description which is not intended necessarily for computer processing, but which is useful when people talk with each other about the capabilities of various systems, the capabilities of various processors. A typical conceptual description might take the form of a graph in which the nodes represent data items, and the lines between the nodes represent relationships between data items. Another level—these are stepping down now, or becoming more and more specific as we go on down. The logical description level is the description with which the application programmer works. The logical description is his description of the data, the way he views the data. It's at this level, I think we would agree, that the data division of a COBOL program falls. The physical description, which is sort of a new description, and which I'm sure we're going to have more to say about, is a description of the specific manner in which data are represented in a computer. I say this is new because in the past, this sort of information has tended to be there and buried or implicit in procedures which operate on a user's program. But I think we're going to see that this description will become more and more explicit. Finally, a coded description, which is essentially a machine language version of the users' logical and physical descriptions. This is the description as it actually resides in the machine, usually with the data, although not necessarily with the data.

I'd like to just briefly run over these questions here. Maybe try to clarify them so that you can respond to them more meaningfully and give us the kind of feedback that we think is important. I mentioned earlier that one of the prime reasons for data description is that it affords a certain degree of independence of the programmer from the physical representation of the data. So it's a meaningful question to ask: Is this kind of independence useful? Is it really necessary? It seems to be sort of a key question. If it is important to have this kind of independence, which levels of data description are need accomplish it, which are the four levels that we enumerated earlier.

Data description languages are potentially, at least, important or useful in the fairly new subject of automatic data translation, which can be defined very, very loosely as converting data from one physical and logical form into another physical and logical form. So the question here, is there a need for this kind of translation in various situations? For example, in converting from one machine to another? I think this question came up yesterday. Somebody posed the situation: Can you pick up a disk pack from this machine, create it with this operating system, and take it to this machine with another operating system? Is there a need for translation in computer networks in which a data is scared among computers of various kinds? Is there a need for translation in providing compatibility among various database management systems and procedural languages? Is there a need for this in processing archival data? That is, data which has been captured in machine language form many years ago, before we even thought about the problems of data description. But now we would like to essentially recapture that data.

Which levels of data description are needed for each of the cases cited in 4.3? Should the data description be stratified in these four levels? Or is there a better way of looking at the problem? Is it possible to work on languages for each of the four levels independently? Or must there be a sequence observed in developing languages? Is there a need for a common data description language in the sense that there is apparently a need for common procedural language? Is the data division of COBOL a suitable base for a common data description language? Finally, should the efforts of the USASI Committee, working on data description languages, be coordinated with the CODASYL work? Our next speaker, John Gosden, is familiar with that USASI work. I would hope that he could say a little bit more about it than I could.

OLLIE:

Thank you, Bill. John Gosden, as suitably primed by yours truly, has very kindly done a little bit more than I expected of him. He has not only prepared his answers to the questions; he's also prepared them in writing, beautifully typed in two type fonts.

Beautiful typing job. [Laughter] I think that what we should do is after John has finished talking, I'll ask Jim Fry and maybe John Young to pass out these sheets so that you'll all have a recording of John's answers. For now, I'll ask John to address these.

GOSDEN:

I hate to get into two fights with my chairman right at the beginning. I'm going to assume you've all read my handout, and start from there. [Laughter] There will now be a short intermission.

OLLIE:

Well, I wanted to be fair to the previous speaker, and I didn't want things being passed out while he was talking, so John, the microphone is yours if you wish to carryon talking while your things are being passed out.

GOSDEN:

Let me give you the background that isn't in the paper. I apologize for not having been here yesterday, and various other things. I will try and assume you do have an adequate background. I will claim to be at least a co, in time, inventor of the phrase "DDL" in a letter I wrote to X3.4. I've been very excited about this problem for some time. Many of you have probably been following me around, as it were. One of the results of that came out of several speeches I've given in various versions on compatibility. For those who are not familiar with the background, see my paper at the Fall Joint '68, which has all the earlier references cumulatively in it. Also about the same time, I got involved with the chairmanship of the X3 Ad Hoc DDL Committee, which is not studying the problem, but studying what should be done about the problem, which is a typical matter approach we have around here. That was the task assigned to four people who sent out invitations to others, and we collect about two new people on our mailing list each week. We're at least up to 50, and the result of that, among other things, SIGFIDIP was sort of formed, which worried about the theoretical side of the problem. The X3.4.1, for those who understand that jargon. We had a number of meetings, one thing or another. Had lots of discussions at Boston with people. As a result of that, I wrote a one-man minority chairman's report on that. I've not circulated this, staffed it, or anything, but I refuse to discuss it as though I hadn't written it. That document will be circulated to the X3 Ad Hoc DDL very shortly to see if we need to have a meeting before I send it back to X3.

This paper answers those questions in the light of that document. Bill Ollie asserts that nobody knows what a DDL is. I claim that's not true. He knows it implicitly, and I think I now know it explicitly, having listened to his paper, the SIGFIDIP, and have translated that into my simple English. [Laughter] You can translate my jargon. I have no glossary established for you people yet. Therefore, 4.1 and 4.2 disappeared. In fact, all these show is that you need to be able to describe completely all the logical and physical data in order to be able to communicate between mechanical systems. The human needs to be able to describe all the logic in the data and as much of the physical as he needs to. For example, if you're sending in punched cards, you have to tell the damn thing which fields it's on. It's no good just giving it the logic of the data on the punched cards. On 4.3, I answered this question in my compatibility paper, and the answer is yes. Now, every one of these is a special case. The answer to compatibility is, if you all do it my way, we have no problem. We all know that that isn't going to be a solution, and they've merely written out four different varieties of people doing it in different ways. I maintain that the monolithic solution approach is the last century's approach, or the last decade, and now we're in the decade of multilithic systems, making independently created and managed systems work together. That's what compatibility is all about.

Question 4.4. Well, Question 4.4, you really have to go to 4.4.1, and this is my rewrite of Ollie's paper. I essentially say that there is a three-by-three—not four kinds of things—is the kind of language. English, if you don't any better. A formal language for humans—you know the ALGOL version that you ship around for other humans to read. And then there's the machine one; that would be whatever we put on the tapes about data description. There are three classes of that: conceptual (that means writing the glossary), logical (just talking about only the logic of the data, which we'll probably never do in a pure form, and then being able to write about the physical as much as we need to, whether it's just the card fields or the layout on the tape. Or whatever. This gives us nine potential types of languages. I've then identified five levels of languages in sophistication all the way from the equivalent in POLs from assembly code to P0s, COBOLs, and the dream towards uncalls of the PL/Is on the way. I assert, essentially, that we need in particular to be able to describe both the joint physical and logical, for transmission purposes, more than anything else. I agree with that statement that Ollie made in his paper. He also asserts we should call it a data description language because a suitable data description would be terrible.

Now, 4.5. Do we need a common data description language? That's like saying, "Do we need a common data code?" If you use the escape clause that Bob Beemer has been explaining to us for years, and most people haven't been listening, I can turn any miscellaneous collection of things into one thing while using the escape clause to exit toward the alternatives. So I maintain we can have a common one and not have it all at the same time, and I don't need to enter into that debate. On 4.6, "Is the data division of COBOL a suitable base?" Yes, I'll say, but it won't cover everybody's thing. We can grow from there. We'd have to grow others. If there is an on-call in this thing, which I called an "UDDL", a Universal Data Definition Language, I don't know if it exists or not, and I don't mind because I think with the escape mechanism, we have a way to get there in the meantime. When it comes to 4.7, somebody has to define "coordinate." We also have to decide whether it's a reflexive properly. I don't really mind, but I'm sure Charlie Phillips and CODASYL do. Did I keep within my 10 minutes? [Applause]

OLLIE:

Thank you, John. Now I'd like to throw this topic out to all. I'd like to remind you, for the purposes of the tape recording—not so much for the benefit of the other people in the room—if you would be so kind as to give your name and affiliation clearly before you launch into your remarks. The first gentleman is over here.

GREENFIELD:

Marty Greenfield with Honeywell. I wonder if John would speak for a relationship between data information, the description of data.

GOSDEN:

Yeah, well, see the IFIP glossary. Let's do without the word "information" until we get the human on the end of it, who makes an interpretation of it. I don't think that there's any line to be drawn there. I'll give the following example. Tell me, in the following sequence events, when data becomes information. I have a series of bits in the machine. I happen to know that that means in Decimal 56. I happen to know, for some reason or other, by table-lookup, that that means the abbreviation "ENG". I happen to that know that that means "country code England". I happen to know that means, on a map, a line drawn around a certain thing. Where do I stop, and when do I understand what that original set of bits meant? When does comprehension begin? It varies for every recognizer mechanism in the system. I'll maintain that it depends on the individual. When he says he knows, he knows. Now, he may not really mean he knows, and he may have to back off to that later. That's how I look at information and data, so I just look at data.

OLLIE:

I'm glad you brought up that point, John. The IFIP glossary. I'm not quite sure how relevant it is to the session. The definition of information in the IFIP glossary is, "Data to which human beings can attach meaning." I think that's a very valuable definition to hang on to. Mary Hollis.

HOLLIS:

Mary Hollis, ISL. I think that John Gosden raised a point that data can be made into information, but rather, it becomes information at the time the user does make use of it. It's not the capability; it's the actual making use of it. In other words, information is very much user oriented. Data always has that potential because that's what we mean by data. But it only becomes information when it is used.

OLLIE:

This is an interesting tangent, but I'm not quite sure how relevant it is to the topic of data description languages. I'd like to ask that Dick Kerrs-- Did you have your hand up, Dick? Okay, Jonas. Jonas Raven [?].

RAVEN:

[Inaudible] to the four levels of descriptions. I always feel that new [inaudible]. I've always felt, as with that time sharing requirement, there's a Council of Mutual Memory, which is always [inaudible]. Also in the data management report for the storage structure, is also [inaudible]. And neither of those terms [inaudible]. I wonder the new concepts here other than virtual [inaudible]. I know it's confusing, but unless you [inaudible].

McGEE:

Bill McGee. I guess I don't really understand what the new concepts are that require new terms.

RAVEN:

No, what I'm saying is that in the time sharing requiring systems, we also have a lot of ??? manipulations today, and there's a term called visual memory, which [inaudible] descriptions, right? Now, the data management report of CODASYL, there is the storage description, right? Which is also [inaudible]. But I'm saying neither ??? ??? storage, compared to the four levels. I'm just wondering why is there new concepts here. Why [inaudible] this confusion was the new concept.

McGEE:

I think the term "physical description" here is meant to include what we have been calling storage description in the committee report.

RAVEN:

So how come you use the new term?

GOSDEN:

What's new about the term physical?

RAVEN:

I didn't say new. I'm saying that we do use it in storage [inaudible].

OLLIE:

I recognize Jean Sammet.

SAMMET:

I have a question for John Gosden, and I request some kind of translation of his data.

GOSDEN:

That's how I expand my 10 minutes.

SAMMET:

I'm aware of that. I'm not giving you that much leeway. On Item 4.4, your answer to the question says…

GOSDEN:

Thank you, Jean. The answers as follows: In 4.4.1, take the phrases in the parentheses and take the initials "C", "L", and "P" as being a prefix to DDL and the initials "N", "F", and "M" for "natural formal machine" as being a post-fix. Now you can read "PDDLF" as "physical data definition language in a formal form." You can read "PDDLM" as "physical data definition language in a machine form."

SAMMET:

Ah. It's a tribute to either you or me or both that I got that far myself. It's the CND thing that lost me. [Laughter]

GOSDEN:

I refer you to Reference 27, that inadvertently didn't get typed, which is Bill Ollie's unwritten, unciteable paper, given that SIGFIDIP, in which he referred to the four things he had at that time as A, B, C, and D. Okay, I've translated it. Unfortunately, Mr. McGee didn't keep the jargon going that he started in Boston, so I'm out of phase and obviously out of date and using archaic conventions.

OLLIE:

This gentleman here.

HARRISON:

Joe Harrison, National Bureau of Standards. I've had a question on the meaning of the term "data description language". I believe ever since I first heard it I'd like to throw it out and see if I might get an answer to this. One could describe data externally to the data itself, or on the other hand, he can conceive the scheme whereby the data more or less describes itself, the most notable example being throwing in suitable sentences that have meaning with the data. Is the term "data descriptive language" understood to mean the four of these, the external, or the self-describing system for it to cover either one?

OLLIE:

Let me talk along there, Joe. What we're trying to say here, first of all, is data descriptive language is passé already, and we're talking about data definition languages. I think that the answer to your question is really both. This is what John is saying in his encoded answer to 4.4, that we need a data definition language which defines the data as it is stored. That language should be something which is readable and digestible by people. We also need a stored representation of that language, which can be stored with the data and which can be processed by machine processors, pieces of software, to find out what that data is all about. Okay? Gentleman from MIT.

BUSHED:

Anheim Bushed from MIT. There are two things that I wanted to comment on. The first was that Bill McGee mentioned about the use of COBOL data division, and whether it is adequate for ??? ??? description. I saw ??? that it is not adequate, and in fact, it may be the wrong track of doing it if you start thinking in terms of COBOL data division for the data descriptive language. For the simple reason that if you want, or if anybody wants, this data descriptive language could be widely used by different procedures. That's including the different users, employing different languages such as PL/I or FORTRAN or any of the others. That it is important that these languages be able to interface properly with this data description capability that the user may have. So for this, to narrow it down to something like COBOL data division, we would be losing track of the proper goals, in a sense. Moreover, the COBOL data division as it is certainly does not include the capability a user would want in such a data descriptive language. Furthermore, there are certain logical relationships with exact specifications, and some of the other primitives. I would like that to be discussed by the panel.

OLLIE:

John, do you want to accept that job?

GOSDEN:

The answer to that question is really quite long. I've addressed it in my compatibility paper under the topic, "Zones of Compatibility", which I maintain that the more areas of compatibility you want, the more people or things you want to be compatible with, the tighter and tighter your rules get, until maybe they get so tight, they're unusable when you have universal compatibility. I'll maintain that within a suitable zone of compatibility, COBOL is adequate. Put what you want, which is perfection. I'll maintain something more is needed. You'll need to define your zones, and I'll maintain we need a whole raft of them from here to there. The answer to it is really quite complicated. There is no one answer.

OLLIE:

John Young.

YOUNG:

One question and a request. First of all, I'm not quite sure how your escape mechanism works. The idea of a common data description language is that the receiving end will understand when it's something over. If you escape into something else, this also has to be understood by the receiving end, in which case it's really part of the language. The request I had…at the UCLA symposium, you gave a very good list of problems involved in data conversion of translation. It would be useful to mention those problems because I think they're enlightening.

GOSDEN:

That was this year's or last year's?

OLLIE:

This year's.

YOUNG:

The fourth generation.

GOSDEN:

The one I sent a deputy to. I can't remember my list, but let me answer your escape clause one. Let's postulate that a universal data description language doesn't exist, but a large variety do. We can register with them-- You have go in the registry business when we have escape clauses. We go into the registry business, and essentially, we have a list of all the general purpose DDL converters. Maybe there's one for PL/Is, and there's one for everything. And you have a code at the front which says which set of rules you're using. There are codes that say, you know, and this is a private subroutine belonging to a community whose registry is controlled by BEMA. You look that up in the BEMA registry. If you're down at the registries, you have to go get them until finally, you can end up with a subroutine that's in a completely implicit data description language that's encoded in the escape codes. Instead of calling in a converter as a table, you call it in as a subroutine, which is guaranteed to do the job and nobody knows why. So you can, in fact, have everything from completely explicit to completely implicit DDL conversion by suitable use of escape clauses and registering them. That's the advantage of standardizing, your registry gets a lot smaller, obviously.

OLLIE:

John.

YOUNG:

If you'll quickly go over the [inaudible] this table.

GOSDEN:

I'll buck that up to the chairman.

OLLIE:

Okay then, John.

YOUNG:

The problems mentioned in this paper…the sender's format may not be specified rigorously in a new form of description maybe to be debugged. The sender's format may not be expressible in the receiver's system. The sender's format descriptions may be embedded in his program. The format in the sender's system may vary from record to record and be embedded in the data.

GOSDEN:

Okay. The answers to One and Three are, I have to write a subroutine to handle those because they're partially implicit. They're implicit cases. The answer to Number Two is, "not expressible" is addressed in a paper I'm writing right now, and essentially goes as follow: We need to know, if we're going from Language A to B, what the subsets are for perfect compatibility. If you want to be able to do it, you have to live within it. We also need to know, from SIGFIDIP, Jean, that if we don't obey those rules, what is the information lost as we go. Once we know that, we have knowledge about what we're doing, even if we don't have control. The fourth one, I've forgotten.

YOUNG:

The format in the sender's system may vary from record to record and be embedded in the data.

GOSDEN:

Well, that means you have to a DDL which doesn't necessarily apply to a whole trial, which is variable from record to record, or even from group to group, or even from item to item. So you need all kinds of data description abilities. We need to have a common data description at the front that applies to everything, or taggable all the way down, hopefully on a default basis. That really says how clever the ultimate DDL has to be.

OLLIE:

Jonas Raven.

RAVEN:

I have the question. Again, a similar question to the question about the four levels. Does the concept or term of virtual memory [inaudible], or is it the same or not?

OLLIE:

Bill McGee.

McGEE:

I don't think there's any connection between the concept of a virtual memory and the concept of data descriptions, as we're talking about here.

OLLIE:

Virtual memory is not movable storage space, and I think our problems come with movable storage space, such as disk packs and tapes. Virtual memory, it's a buffer station. Nobody's going to take your paging drum and ship it somebody else. That's really why I think that this virtual memory thing is sort of a…

RAVEN:

But then [inaudible].

OLLIE:

I think you're asking the same questions we had down here. It's the external appearance of the language which defies how the data is stored, the language, which is for human consumption, and the stored representation of that definition, which we see in most of the generalized database systems we've been studying.

GOSDEN:

Let me try another answer, though. Let's assume I'm in the machine here, and here is the virtual memory interface. Okay? And essentially what happens is, somebody knows what's going on over here, but what virtual memory does is give you a way of talking about it; in the same sense, stick to that. Now, I will now talk about two specific varieties of the machine data description. On this side, the virtual memory is transparent to it. But when I passed it through here, it would know about this side. In fact, in that system, there will be two varieties of the machine data description, depending upon which side of that interface I was. But this one doesn't know whether that's a virtual memory interface or not. It merely asks questions of it. Everything it says is stated in the terms of the environment in which it inhabits. When it goes through here, a translation occurs, and this one states it in terms of the environment in which it has it, which may be through an operating system, and has another translation all the way down. So the answer to your question is yes, when it is taken into account depending on where you are. Every language must be bounded by the environment and the knowledge which it has, and what it can talk about. Does that help?

RAVEN:

[Inaudible].

GOSDEN:

See, this paper is going to be an encyclopedia when we've finished is the trouble.

OLLIE:

Yeah, I think one of the things we're saying here about data definition languages is that we're still groping for directions. Work ought to be done, there's no question about that. From a CODASYL point of view, we have the question, and it's really Question 4.7, and maybe it's a political question rather than a technical question. It came out yesterday fairly clearly that this data definition language thing is an important thing. USASI has John Gosden's committee, and the ACM has the Special Interest Committee. Both John and I, for example, are active in that committee. What should CODASYL be doing about this? Would any of you like to make any comments on that issue? Charlie Bachman, I think.

BACHMAN:

I think one of the major activities, which will probably come in the next storage presentation [inaudible], that the Database Task Group, a considerable part of their activities and worked on data description languages in order to describe that database. This doesn't cover the entire ???. It covers one area quite thoroughly. Should be a level of [inaudible].

OLLIE:

That is a very pertinent comment. I know I have seen tomes from the Database Task Group with the label, "Data Description Language". The thing that they are producing is certainly something, as Charlie points out. On Level B, it is a logical description, but I think it also has a flavor of C, the storage description or the physical description. There's an open question, for instance, as to should one separate the logical description from the physical description, or should one merge the two, to the confusion of all concerned. John?

GOSDEN:

I would like to give a credit where credit is due. Your question is good, but slightly at the wrong time. We've been floundering on this for about a year and a half, and the definition of what it is and a way to talk about has been most elusive. The paper Bill Ollie presented at SIGFIDIP, which just took a first cut at the problem, really divided it up into four classes, which I've redivided into nine. You're saying, why haven't we done the classification scheme all the way down? Wherever Bill Ollie is giving me credit for recognizing something in here, it's mostly because I rephrased what he said. I encourage him to publish his paper and send it to you for extrapolation to the next level.

OLLIE:

John, I didn't want to argue about the Four and the Nine. I think the one you're missing out in your Nine is the one that-- in your three-by-three. I agree that there is a second dimension to the taxonomy. But I think that the one that you're missing out in the dimension that is compatible with the one that I gave in Boston is the Level A, the conceptual description, which is the one that we felt a need for on the Systems Committee in our survey work. We would like some formalism for defining the degrees of freedom that a database system has. Now, this has nothing to do with communicating with machines. It is a formalism across a set of systems. Well, you threw that at me two minutes before the session, and I didn't have time to look at that. Do you have any other further-- The lady in the back row, please. Could you give your name?

BETTY:

[Inaudible]. Back 10 years ago, on the data description [inaudible], this logical/physical was a really ??? argument at that time. The reason for going to the logical was nobody has a change at that time. [Inaudible] was it a round hole, a rectangular hole, whether they had to assign ??? in one place and their data the rest. Nobody had a change in order to get ??? ???. If you're going into a data descriptive language as you're talking about, it seems to me that there had better be a consensus of the industry. Yes we already have a change, and a major change, in the way in which we're storing our information. Before we even get down to the rule of what it will be that they will have ??? ???.

OLLIE:

That's a very good point, Betty, and I hope that other people here agree with that.

GOSDEN:

I don't. I think we have to live with the current situation as well. You have to phase things out and to try and just put down a perfect solution--

[CODASYL Disc 9]

GOSDEN:

--shuffle over to it is guaranteed to bring disaster and riches to all the EDP industry.

OLLIE:

You can't get there from here overnight, and I don't think that's what the lady is saying. But you have to know where you're going.

GOSDEN:

Oh, yeah, but that goal may keep changing. As you march down the road, you might suddenly find you're doing local hill climbing and need to backtrack and go up somewhere else.

OLLIE:

But it's interesting to note that these problems were here 10 years ago.

GOSDEN:

Yeah.

OLLIE:

Mary Hollis.

HOLLIS:

Mary Hollis, ISL. I'd like to point out also at the time of coming up with the data description language that we used in COBOL. The reason for coming up with a data description language was that more people in a data processing system have to use the same data. In other words, there are many programs written using the same data. It's what is necessary to have this data defined or described in the certain manner. Now, I'd like to change in the terminology from "data description" to "data definition" because I think it recognizes that as we go to a database system, we have to deal with many more people, and that's including users, systems programmers, and whatnot that will use this same data. Therefore, when you describe something well enough, there comes a point that you can now that data is defined. I think it recognizes that we must take this step. CODASYL is we're supposed to do development work. This is one of the reasons that we are in existence, to do development work. It makes little difference to me where this work is done. The main thing is it must be done, and ??? ??? to do it.

OLLIE:

Mary, I agree with you wholeheartedly. The thing I'd like to say publicly, too, is I'm a technical person, not a politician. I think this work ought to be done, and I really couldn't care less where it's done where it's done as long as it is done, and I think that's a very important point here.

HOLLIS:

And the fact that we're behind.

OLLIE:

And we are 10 years behind of it, yes, Mary. Do we have other comments from the floor, please? Yes, the gentleman-- No, not you. The one behind him. He beat you by a millisecond.

MALE: George ???, [inaudible]. We've been talking about the different levels of systems. One of my things, it does need some separation physically. There is a certain amount of physical implicit in the data description of COBOL. The size of the field--

[CODASYL Disc 10]

MALE:

--higher specification. After it's in the standard, it's not particularly palatable to anyone and has to be reworked for two years after that. The problems involved in specifying a language that is machine-independent, and of solving the problems that come up among the various implementers of software, not necessarily hardware, makes the job a little more difficult, and you have to exercise a little more caution in getting your job done at home, as was mentioned before. Certainly, you're home, getting the job done. Even the implementers themselves find no difficulty in putting those kinds of facilities in their own in-house languages that are generated for their computers. The problem here is to get everybody to agree, and you can do this; you can force them to agree. If you're a good committee psychologist, you can get them to agree to something by beating them, by making them stay until 4:00, or something like this. The problem is, after they agree and after it's specified, it's not implemented or it doesn't work. This is what we have to be careful of. That's my comment.

JONES:

Thank you. George Mann?

MANN:

George Mann, UNIVAC ISC. I'd like to draw out a couple of threads together. You didn't get an answer, I don't think, on the question about job control languages. I think it is something that CODASYL really needs to do. There is even some hint that there might be some overlap between job control languages and whatever you call the one we have now. I think that Beemer's suggested it, or one interpretation of what he said to me was we won't know which processor we're going to invoke until we get to the entire division in our string of independent computers. That's one of the features now is job control language is a little bit deeper into the program, if you don't know the environment and what we're going to do. I think he's right and I'd rather support his suggestion, that a very interesting way of looking at the problem of getting the languages together. I thought of it again when this gentleman brought up using subprograms, linking. We have a good IO system, then we can call FORTRAN or ALGOL, or hopefully LISP for a few other things down here inside of it, perhaps two or three of the same program. We're doing this today with FORTRAN and COBOL in many places. I'm sure others are. But there are some difficulties there that I think CODASYL also can address in improvements to COBOL to make it easier to work with FORTRAN or something else under it. We have in ours already common stories that helps us to get data moving. They're calling ??? as a FORTRAN option on a particular COBOL. But doggone it, we don't have a floating point description in COBOL, and that's a pain. I think that CODASYL can address itself in that problem. Segmentation? Who cares whether the other segment is COBOL? I think this hasn't been very well defined.

JONES:

Thank you. Before we get going, I'd like to raise a question as to what opinions anyone might have in regards to this kind of a meeting, and when, if ever again, such a meeting should be convened. There's a possibility of waiting till the 20th anniversary. There's a possibility of trying to meet on a more frequent basis to reaffirm or readjust the plans that will undoubtedly develop from this meeting. What thought does anybody have as to this question? For example, I happen to feel this meeting has been very valuable, very worthwhile to us. We've gotten a lot of guidance from it that we'll have to sit down and try to distill. But nonetheless, I think it's been quite an enlightening meeting. When, if ever, would it make sense to do this again, do you think? Charlie Bachman.

BACHMAN:

It's probably too soon to know when the next meeting should be had. I don't think it should be an annual affair. I think you have to wait, and those circumstances kind of say that it's about time to have another one of these. So I propose to kind of say, well, we had a good one in 1969, and the next time it seems appropriate, then schedule it. I don't think you can plan one as being appropriate five years from now or two years from now.

JONES:

That's essentially what we did this time. It looked like this was a good time to talk about some of these problems. Mary Hollis.

HOLLIS:

I feel very definitely that when is meeting is planned, then you're supposed to have certain amount of work done by that time. I think it's a tremendous incentive to get that work done. I would suggest that we schedule the meetings and schedule them on a rather, say, annual basis. We expect certain work and certain reports to be available at that time, and I think we might get them.

JONES:

Well, I do know there was some scrambling to get ready for this one. Bill Ollie, isn't that true? [Laughter]

HOLLIS:

I think there should be more of them.

HUBERT:

Hubert, DF Goodrich. I'd like to address a question to you. In your estimation, is CODASYL going to be more active than they have in the past, less active, or about the same?

JONES:

I guess I'm really not in a position to speak for the entire organization because the Executive Committee is going to get together and try and distill from this meeting what really has been said to us. I think it is my impression that, in general, we've been told to get on with the job of building some certain pieces on COBOL that need to be put on there. We've been told to do a better job letting people know what we are doing. I think we have perhaps been told to get on with the job of the data management/data descriptive business, and get that better defined as to what we really are talking about and get on with doing some work in that area, and perhaps other items as well. I'm just reciting quickly a couple of the ones that come to mind here. I would say that probably my own personal opinion at this point is that we've been told to get more active. That's subject to confirmation by my peers. Yes, sir?

BUSHED:

I'm Anheim Bushed from MIT. I would just like to make a comment on something that we've discussed. I do feel like if you do intend to become more active, that it's some kind of a support coordination, especially from the full-time staff as well as staff may be, you know, which would in addition to the community that you have, which are very useful in giving direction, but somebody who will be able to take some of the immense amount of the work of coordinating for sending the tools and calling the meeting, and all these kinds of administrative functions, because that really bogs down many people in order to, say, give the minutes of the meeting. It is a thankless job to be sitting there and be writing reports on the rest of the meeting or distributing or just doing things of this sort. I think maybe it's [inaudible] the support some of the ??? to do this kind of thing.

JONES:

Thank you. I know I can speak from personal experience that I don't get much of my paycheck for doing CODASYL-type activities. There are times that it's just impossible to do the best job that could be done, and just because of that conflict. I certainly think you have a good point. Tax?

METAXIDES:

Tax Metaxides, BL. I was wanting to comment about your summary that you just made. I feel that if indeed this meeting succeeded in getting across some of the thoughts that you summarized, then you have to ask about how often. I believe we should not wait another 10 years for offering direction, and I would suggest that most business organizations have some kind of board of directors; that we have a planned annual meeting whereby some information can be interchanged between the executive and the members. [Inaudible] and inputs. I propose that it is a two-way street. I feel very strongly that every 10 years is far too long. If you try and fix a number, well, it seems to be that an annual meeting would be about right.

JONES:

Thank you. Charlie Bachman.

BACHMAN:

One issue I thought was appropriate for this meeting, which I have really held off of discussing, is that I think something significant happened at the New Orleans meeting of the Programming Language Committee. A document of some 40-pages magnitude is adopted as an extension to the language, then almost nothing is said about that extension in terms of the message control systems language. I think it's very appropriate, before we go away, that we recognize that a real important extension has been made to COBOL just recently. We talked about data descriptions and we talked about data management. We haven't talked about ??? theory. I think it's appropriate that we have some discussion of that topic before we quit. But this is a step forward. I think it's not actually as far as we'll go in this area, but certainly a major step forward allowing a program to talk to a terminal, which is not generally available in COBOL.

JONES:

I think that was a very major extension, though; there's no question about that. Are you suggesting that, perhaps, Ron Hamm should give a five- or ten-minute discussion of the concepts that we have added to COBOL and this extension?

BACHMAN:

I think it'd be very appropriate.

JONES:

Would you like to have that? [Applause] Ron, accustomed as you are to public speaking, I'd like for you to come up. You can organize your thoughts there. Walk slowly and organize your thoughts. [Laughter]

MALE:

Pull out that 20-page speech, Hamm.

JONES:

I think you're aware Ron Hamm is from Honeywell. He has been chairman of the task group that worked long and hard, and I know they've put in a lot of time and effort because one of my fellows, Dick Tripp who's here, participated on this also. In fact, the communication extension may have been some sort of a record for the elapsed time between when it was started and the time it was passed in relation to the magnitude of the change, whatever that means. Ron?

HAMM:

This is what I call short notice. It's a Herculean task to describe the communications facility for that. It took us 18 months and a tremendous amount of work for many companies. I hesitate to kind of give a tutorial. One of the things I'd like to say about it is what our philosophy was when we started on the communication facility. I have listened for two days, and many of the things people have said, I'm glad to say that the Communications Task Group thought of. There are two that come to mind, and I encourage the other members of the Task Group to chime in. We felt strongly about not dictating what happened in a particular operating system in order to handle the communications facility. The best we describe ourselves is describing the behavior of the message-processing program. That's part of the job of a data processing. It belonged to COBOL. We felt it was to take the message from an operating system (we called it MCS, a message control system), and not dictate what happen in terms of queuing, polling—all the other communications functions that we lumped under the communications discipline. We felt that the COBOL job was to interface with it. We defined an interface. We certainly admitted there were many inherent functions that had to be supplied in an operating system in order to support this language. Much like Tax said, if you don't come up with the operating system and the vendor software to support it, it's syntax without meeting.

I think we're very conscious of the fact that COBOL, and certainly not the Communications Task Group, was in any position to dictate a communications facility, per se, for COBOL, and then expect a vendor to support it, and also one for PL/I, for FORTRAN, for their assembly programs. We are very, very conscious of anything. We've spent a considerable amount of time making, as sure as we could-- Again, I'll be back, I hope, in five years, and that'll be really the test of our success. We made a first crack at it. We tried to stay out of the operating system. We tried to define what we needed in order for the COBOL program to do data processing. We identified what a message was. We identified some of the basic characteristics and information about the message that we felt he needed in order to make his data processing decisions. When we got into the messy area of queuing, polling, line discipline acknowledgments—many of the things that the USASI X3 Committee is doing. We may have taken the chicken's road out and said, "That's their job," but we also felt that we were supplying a facility for that applications programmer that Dick talks about.

That's Major Philosophy Point Number One. The other one was to be terminal-independent, hardware-independent. It doesn't take long for new people just to sit here and compile a list of the number of different and varied terminals that there were. You can think of, probably, many more than even the Task Group has come up with, but you have such diverse things as another computer, another program within the same program or another computer, and then you have the traditional on-site devices like tapes, card readers, card punchers, and then the teletype CRTs. That's a bag of different devices, and we felt, again, that the basic ten of the CTG philosophy was, this programmer couldn't care less about the particular characteristics of that device. He was interested about the message coming across from a device. He was very conscious some logical information so that he could identify the sender, the terminal, but he couldn't less really if it was a 1050, or what have you. So he tried the main thing in that throughout the proceedings of the Communications Task Group. I hope to publish the documents as soon as possible to the community and invite your criticism, certainly. I would like to not go through another task group several years from now. Many of the fears that Dick's expressed certainly weighed heavy on my money during the chairmanship of that task group. That's, I think, what I could identify as the two major points.

I think there was an auxiliary function that was taken on in CTG. We felt, after examining the things the communications COBOL processing programmer wanted to do, he needed to work with raw data: unformatted data, unpredictable data. And we felt it rather cumbersome to handle unpredictable data with what I think you would do with the redefined SQAZ [?] in COBOL. There is no real good way to handle strings of data, followed by delimiters. We added, as an auxiliary to the proposal; a good portion of the proposal adds two new verbs to the language—in no matter, shape, form, or communications, per se—that allow you to handle strings of data independent of the size or description that you've given it in the data divisions. Hopefully we're string processing, and we call them string and unstring. That was what, I think, a very beneficial byproduct of the Communications Task Group.

I hesitate to go any further into the details. Perhaps if you ask some questions or suggest some kind of facilities we may have considered, and I did invite my task group to come to my rescue. Yes, sir?

WRIGHT:

Steve Wright. When you say you're going to publish it to the community, what do you mean by that?

HAMM:

Well, that's yet undecided. I think the SIGPLAN is certainly open to us. I think the Database Task Group published by registering the document with the clearinghouse.

Tax, perhaps you can correct me if I'm wrong. And then making it available, making announcements of it available. It is available as page changes, I think, to the document very soon. I think the Canadian federal government, again, is making page changes available to you. So those of you, I think, get a right to the Canadian federal government and get a manual. You'll also have the opportunity to get current page changes. Dick says we make changes every meeting. We, in fact, do. They are reflected in official change pages, which you can keep quite current with.

BUSHED:

Does anybody have those copies of any of the task group manual or something like that? Like the document? Where could one get it, by writing to whom?

HAMM:

I hesitate to say write me because then all hell breaks loose. But I will, as soon as possible, get it out, and there should be announcements in several publications. I understand Mr. Simmons has an exhaustive list of places we've been published. That kind of announcement—where it'll be available. Certainly, I recognize the need for getting it out as soon as possible, and I think it's a two-way street. As a task group chairman, I literally tried to beat the bushes of trying to get people to come and sound off on the proposal. I think one of the things I have to take my hat off to is that the companies and the people involved in the Task Group really pitched in, and I think that's the success. There's the keynote for success to the Task Group. I think we, as members of CODASYL working on this Task Group, are interested in your comments. We want them early, and I think it's really one of the things we'll try to implement, I hope. [Applause]

JONES:

Thank you very much. Of course, you always have this same problem, and that is, until you specify it, it won't get implemented. But as soon as you specify it and try to implement it, you're going to find out that you didn't specify it quite right. So it's going to have to be changed as it goes on, and I think that's just a natural course of events. Any comments or help that we get prior is just that much to the good, of course.

Let me remind you once more that we'd like to have your questionnaires. I just don't want you to be able to say you forgot. Do you have any other comments, questions, criticisms? You've been waiting all for two days to sound off and blast us. Go ahead. Bob Wickham [?].

WICKHAM:

I stated before that this is my first attendance at one of these meetings. I've enjoyed it tremendously. I think a great deal of credit is due to Jack Jones, primarily, the committee chairman, and the other committee men for organizing this and getting it together so we sound off. Can we have a little hand for them? [Applause]

JONES:

Thank you very much. Of course, the two real workhorses in our organization are Dick Kerrs and Bill Ollie, who have, for some time, been heading their committees. As you can tell from their participation, which was almost continual in one form or another, they are the real motivating forces in terms of the products in the sense of COBOL and database survey and so on. Yes?

SWEENEY:

Jim Sweeney, UNIVAC. Jack, you made mention earlier that due financial constraints, you did not intend to publish the proceedings. I do think that the people will be interested in knowing, is there anybody by which they will receive some indication from the Executive Committee as to what you gleaned from this meeting prior to that next meeting 10 years from now or five years from now, or when it will be. I think, perhaps, one of the ways would be, if the Executive Committee were to furnish such information to PLC, as an example, then it would be reflected, perhaps, in the PLC meeting minutes within the next couple of years. But what vehicle, if any, is there prior to the next meeting, whenever that is?

JONES:

I certainly would be reluctant to promise to do something that may not come about. It would reasonable that the Executive Committee will be able to come up with some summary or some synthesis of what's been said and what we interpret from it. Maybe if for no other reason than to send it around and get you all to write back and say, "No, no, you didn't understand." Let me say I think we will try to do that. I'm not going to promise that yes, by gosh, we will, because maybe we'll sit around and scratch our heads and are not able to come up with something, but I don't anticipate that to be true. As you know, the session has been recorded. I think I announced to start that the Smithsonian has very graciously offered to transcribe it. It may be that when it is all done, it can be registered with the clearinghouse and the proceedings of this meeting could be available to that means or something. We will explore that, certainly. It's certainly not the intention to keep all this secret in any way.

If you have no other comments or questions, I cannot adequately express how appreciative I am and the Executive Committee is and all of the actively participating members of CODASYL are for your coming and your participating and your guidance, because this is precisely why we got together. I think, in general, we have had a good dose of what we asked for. I hope that, sincerely, we can respond and keep you informed and continue to receive your help. Thank you very much. [Applause]

[CODASYL Disc 11]

GREENBERGER:

Especially the bigger ones, who can support these activities. We tend to send systems programmers to these meetings, and they're not really users. That's a lot of what I've heard today, and that's my reaction, and it kind of distresses me. We've got to get some applications, and we also have to get some ultimate customer user feedback into this system. So I think we ought to recognize a little change in our organization and our people. With all this gloom and doom, I still like CODASYL, and I think it's my best bet to do something about this area. I think this area is the winning area. If something effective is done, that organization's product will be the one that's accepted by the field. Though I did like what John Gosden said, and said it well, if CODASYL doesn't accept what I say, I back USASI and what they're doing, because that's closer to what I support. But I think that CODASYL has the best track record, and I tend to support the horses with the best track record. They have the shortest odds. So that's why I've taken the trouble to even come here. And I'd hate to see CODASYL vanish.

What I think CODASYL should be doing is looking first at interfaces. This is a different priority than Tax talked about. Talking about the interface between the data manager the operator system, and between the data manager and the procedures, and the between the data managers and the communications. I think it's criminal that we software people haven't modularized these things. We've knocked the hardware people, but they now build standard I/O interfaces so we can plug one manufacturer's disk drive on another manufacturer's mainframe, and printers, and tape units. And damn it, we don't have-- No, if we modularize, we can have a software house develop the modules. But we've got to do work on the interfaces. If we go off charging and extending COBOL first, that's a trivial problem after we have the interface. You can do it. You can extend your COBOL procedures to interface. The problem is defining the interfaces. I don't like so much talk about language. I'm talking about functions. As has been talked about earlier, the syntax and the semantics of the language are not important, even in COBOL. People who have talked about using across machine lines have machine-translated, change statements, and flag those that can't be translated. We accept that as an operational definition, a simple program of some sort to translate. So I don't think it's important to specify a rigid language, but I think it's important to know where our interfaces are and what are functions are within the interfaces. There may be finer interfaces and modules on what I've talked about.

Then I think we might go on and talk about some loop lists and well-known data management facilities, like sequential organization and sequential organization, and then specify those in a standard way, but I consider that distinctly as second order of business. So I think that's some food for thought, and I'll stop. [Applause]

MODERATOR:

Thanks, Chuck. Before I ask for your comments on these questions, I can't resist adding one other problem to the pile. Most of my effort in recent years has been not with management information systems and not with databases, but working on the management system itself and determining what information is needed for planning and control so we know what to put in the management information systems. One problem that comes up repeatedly in this area, and it ties back to a comment that Tax made. Tax says, by definition, management doesn't know what they're going to ask for; therefore, the interrogations are somewhat uncertain, and therefore, the accessing of the database, the accessing load and characteristics are really not well defined. Nor is a scenario from a terminal well defined because you don't know who's going to use a terminal next. So the accessing patterns and loads and so forth are pretty uncertain. I think this is the point that Tax was making.

Now, if you talk to a systems programmer about this programmer and start off with that sort of definition, "Management doesn't know what they need," he says, "Well, how do you expect me to design a reasonably efficient database to serve his needs?" Of course, this gets back to the so-called data independence. I gather that's a pretty fuzzy term around here. But there must be, if we're going to move ahead with large management information systems, as Charlie has said, we need to have many people who can get at the thing without upsetting the other fellow. But we also need some way to reorganize the physical database to improve the throughput after we observe experiences on what access patterns we're having. Only through acceptance of that principle can we move ahead with designing systems with any confidence that we're going to be able to change it if the access pattern changes. And certainly, most of you know in deciding what management needs, the minute you put in a new manager, he has a new style, he wants something different, and the whole access pattern is different. So I would like to just add my appeal that we not overlook the problem of physical reorganization of the database to improve throughput. Now I'd like to throw the floor open for questions. John?

YOUNG:

John Young, NCR. We've been talking about various classes of users, and I wonder-- I hate to reuse a concept that I talked about with a difference of application, but I wonder if don't have to look at a continuous spectrum of users, ranging from the absolutely new nonprogrammer, whom you have guide in a very, very tutorial way, up to the database administrator. I think that we will find this new guy learns more about this system and more and more, and eventually comes to be able to use some of the more sophisticated facilities, and you're going to have assistant database administrators who don't have quite as much access to this system as the chief does. Now what I'm getting at in this approach is, I think that we have to start with the hardest problem, namely, the least experienced user, and build up from that. I think once we satisfy his requirements, we can then provide more and more facilities for the more experienced users. Can we get to this final, continuous shading of sophistications of the language by starting with COBOL, which was designed with quite a different end in view, and adding things onto it? I kind of doubt it. I think we have to start with the basic idea that we are going to design a compatible set of languages for this whole group of people, and then go on to the design, rather than trying to catch up something that already exists for a different purpose.

MODERATOR:

A brilliant comment. I'll turn it over to Bill in a moment. I want to make one observation. I gathered while the presentations were being made-- I heard Bill Ollie say that COBOL was evolving in the direction of the systems programmer. I heard Tax and Dick say that COBOL was evolving in the direction of the application programmer. I think it comes back to some modification--

OLLIE:

[inaudible].

MODERATOR:

You didn't? I misunderstood you? Well, do you want to comment on this particular question, Bill?

MALE:

No, my comment was that it was intended for him. There was no evolution involved.

MODERATOR:

I see. I see. I stand corrected. Are there any comments on this point?

MALE:

I think Dick and I feel it a revolution rather than evolution is needed.

METAXIDES:

I see it a perfectly reasonable and respected approach that we have to have that hierarchy of languages. I think the question is where it's going is what's being discussed at the moment, and I think that rather depends on the person's orientation. I saw a series of maps of the world from different points of view. That is, the world as it looked from

China, or the world as it looked from India. It's surprising to see how very different it looked. This one is involved with a particular-- is one kind of a user, then one tends to look at the whole problem from that point of view. I personally feel that for the nonprogrammer, the fellow that needs tutoring, he also needs data. My reason for considering that the stop point should be the more procedural aspect is that how else do you create the data if it is not a normal byproduct of your day-to-day operations? It's wonderful having the facilities, but no data.

MALE:

I'd like to ask another question. I haven't heard any comments towards what we're going to be working with in the way of, just arbitrarily say, fourth generation equipment. Nor have I heard any comments on the economics of why it's very costly to do these things, which are clear and rather distinct ??? ??? our method of a variety, say, database management and whether this is an abstract thought, whether it would be easier to do so-called recompiled each time instead of cataloging the library situation and things like that you could have now. Or anything in this area, and yet I feel that this is going to have a distinct effect in the next couple of years on our whole method of applying on getting into these problems with database managers and things of this nature.

Now, I'd like to hear comments, if they're not all confidential, on what we're going to be looking at in the way of equipment in relation also to the way of economics. For all we know, may end up to the point where everybody's going out on a small 64K computer that they carry in their briefcase, which is completely going to change our approach to languages and our whole approach to database management.

OLLIE:

Yeah, I'll put on my manufacturer's hat. I can't help feeling this question is somewhat irrelevant. You seem to think that the rest of the discussion was irrelevant; I think your common was irrelevant. I think probably the onus of the economics does lie with the manufacturer and with the software supplier. We think that the cost of implementing software systems, it's going up. The cost of the people is going up and the complexity of a system is going up. Now particularly, as we move to online systems, time-sharing systems, again, the cost of the thing is going up. I'm afraid, with no question about it, this is reflected in the economics of the business, and it's the business of the people that are supplying the stuff. I don't think it's the business of CODASYL. I'm sorry.

MODERATOR:

Tax?

METAXIDES:

You some to infer there's some business between the small and the large user. I feel that data is data, and even relatively small companies have a lot of data that is interrelated. Essentially, what we're talking about it is, to put it very crudely, input/output facilities that should have existed all along.

MALE:

I didn't mean to infer it was your role. I was trying to draw another factor in here that [inaudible], whatever the tact CODASYL takes or whatever cost that is.

MALE:

I would like to make a general observation. Again, I'm kind of up on the idea of the two days, and most of us declare ourselves, in various ways and various times, to understand the differences between philosophy and practice, the differences between the bigger view, which we must have to be in a position to at least advise, if not guide, an organization like CODASYL. I would like to point out two comments made by speakers on this panel as examples of the fact that ??? ??? points of view I think is accurate, except to say that we failed in using them. Because the reason that those facts look different is because they are not an accurate reflection of reality. Namely, they're not globes. If they were, they would be pretty much the same because they were accurate. The same thing, I think, exists here. Our problem is not—and I think this is where your analogy falls down—our problem is not the lack of existence of an objective reality. There is such a thing. Our problem is that we're still burrowing around in the dark, trying to find it, to find what the real shape of the problem is, in which case it will, if and when we do this, pretty much look the same from any viewpoint. Point Number One. Charlie Bachman's comment, I find another kind of trap. He made the comment that we cannot allow ourselves, the companies, to be parochial enough to force people to use COBOL when they consider FORTRAN, as an example, as their native language. I believe rather that we cannot be parochial enough as companies to allow people to use FORTRAN for applications, which are more appropriate than to solve in COBOL, for instance, to begin with. And I think that that's the other trap that even we fall into. What's there tends to determine what we're talking about now, and we should not.

MODERATOR:

Mary?

HOLLIS:

Actually, I think the thing is we need to be sensible and talk about approaching any problem. The problem that we are on today is different from the problems that we think 10 years ago. I think that when you're approaching such a problem, the first thing you do is to set down the requirements of this problem. You have to be efficient to achieve.

And this must be done in terms of the end result, which is the user of the system. What are these requirements? Until we do that, we will not get started in the right direction. Because as you well know, if you a standing appeal, and then you were trying to add on, you were adding on to improve what you can, which may not be the answer that you're looking for this particular set of problems. I happen to believe very strongly in COBOL, and we are using it extensively. But I don't think that COBOL, which was designed to serve a certain set of requirements--The requirements today are different from the requirements then. If you extend COBOL enough, if you take care of our database systems today, just forget it, it won't be COBOL. It will be something else by that time. And I think we must look, first of all, at the requirements, and we should get a write-off, so to speak, or get feedback from the users whether or not--This is a set of requirements. And then we set about using that which we have to answer these questions, and that which we don't have, we develop.

MODERATOR:

Thank you, Mary.
.
HOPPER:

Mary, I'd like to [inaudible]. I'm not so sure the requirements have actually changed, so that the users are now getting back to their requirements. Back in 1947, I can remember, distinctly, Prudential saying they needed 500 reels of tape all in a line with automatic threading. At the other end of the line was a two-minute interval ??? ??? find out about policy. This was something they needed. However, they backtracked, and what their real needs were changed the user, changed the persons whose data were being manipulated, to conform with what was the reality then. Now they're getting to the point back all the [inaudible] quickly you answer the call. We'll get the material quickly from the point of view of [inaudible] getting a rough draft. Now we're really getting back to what the major requirements were a long time ago, and doing it mechanically.

MODERATOR:

Tax, you had a point to make?

METAXIDES:

Grace, you made exactly the point that I wish to make. It isn't that our requirements have changed. The requirements are there; we're just now getting around to them. After all, again I stress that COBOL had the rudiments of input/output there, and all we're talking about still is input and output, getting the data you need, and storing that data. I think a rose by any other name is just as sweet. I don't care what you call it. What I'm interested in is having the facility. I don't care whether it's spelled C-O-B-O-L.

MODERATOR:

Mary?

HOLLIS:

I wish to come back slightly to say that the requirements of business today are not the same as they were 10 years ago. With the requirements, our information for running a company today are far more short. We have to have information, and we have to get it in a smaller interval of time. We don't have the money to spend in a development that won't meet our requirements. The requirements have changed because the business environments in which we operate have changed, and therefore the pressures are greater. Now, what one wishes and what are requirements…maybe this is our problem in semantics. But I'm saying that the requirements today are far more severe than they were 10 years ago.

MODERATOR:

Grace?

HOPPER:

I think we're all enwrapped in the same two questions, it's the lowest and the original COBOL. We do need a way to extract data and refine it, and we do need a language to be adequate. We're asking for the same thing, exactly the same thing. Admittedly, when we started doing it, certain concepts weren't acceptable in the industry at that time. And you're right, it could be; then they said they couldn't be. Only then [inaudible]. We still need to talk about what we're talking about when it isn't [inaudible].

MODERATOR:

Thank you, Grace. Bill?

OLLIE:

I'd like to comment on that last remark, and I think it's very pertinent. But I do think two things have happened in the last 10 years. One is that we have a greater acceptance of direct access devices, and we need more complex data structures than we saw the need for, or we even were capable of understanding 10 years ago. So we need to have a way of making more effective use of these things called direct access devices. Point One. Point Two, we need language to get at that data. I feel that over the 10 years, we have developed a need for a language which is less procedural than COBOL. The basic things are still there, I agree with that, but I think that there have been changes in the complexities of the data, as substantiated by the storage devices available and by the requirement and the expertise of getting at the data more expeditiously.

HOPPER:

Right. All I was saying was that these two areas still need [inaudible].

MODERATOR:

Jean?

SAMMET:

I want to make a couple points. First of all, I would be inclined to agree with Betty Holberg [?], but I disagree with Mary on the issue of the requirements. I think the requirements for fast data were needed 10 or 15 years ago, but nobody knew how to get them so people learned to manage companies without getting them that rapidly. Secondly, I would like to pacify Mary by agreeing with her that if the equipment does stop onto COBOL, it stops becoming COBOL. It becomes unwieldy and not usable. There is at least one hysterical example for this, but I don't want to get the issue sidetracked. One thing to point out, that when it worked on what is now called PL/I, which we started, we were going after it by SHARE and by IBM. The original intent was, in fact, to extend FORTRAN to bring into it the character handling and other kinds of capabilities that we felt were needed. The people who were doing that work eventually realized that there was just a limit to which they could push with FORTRAN. You couldn't add on just all the stuff that was needed to do all the stuff that was important, and therefore they had to, in a sense, say, "All right, we will not maintain compatibility." Now where that level is with regard to COBOL, I'm not prepared to say. But I think there is a point--

[Recording jumps to another point in the meeting]

JONES:

Typically, first of all, talk about any thoughts or comments that individuals may have in regard to, specifically, the future role of COBOL. Then get into, again, a more general free-for-all. Perhaps a continuation of some of the talk that had to be sort of terminated for lunch this morning. Try and again draw any further comments out. We have some points that have been covered adequately several times, and perhaps some that haven't been talked about. We'll try to bring those questions up and get your response to them.

I do wish to ask that, again, that you fill out and leave the questionnaire that you received when you registered. We're very eager to have them and very interested in your replies. To start this session, I'd like to ask Dick Kerrs to start. Steve, do you have a question?

WRIGHT:

Could you give us a grace period on the questionnaire?

JONES:

Well, we hadn't planned to do that because we were afraid you'd forget about it and not send it to us. We are more interested in good answers than quick answers in the sense that, I always say, "When you want it bad, you get it bad." But we were concerned about was-- I know how it is when you go to the meeting. You leave with all good intention of sitting down when you get home, and then good gracious, you've been gone a couple days. You put it aside, and pretty soon, the stack is a little higher, and you come to it a week later and you say, "Well, I can't do it now." And then by that time, you sort of forgot what your comments were. So we're very eager, while they're still fresh in your mind, to capture the questionnaire. I don't think we have it pre-numbered in invisible ink, so we wouldn't probably know it. If you do go off, please do send it to us because we really are very interested in the comments and answers. Dick?

KERRS:

I've been on enough of these panels now that if I had to give myself a COBOL data name, it would have to be "Filler" at this point. [Laughter] But I would like to take, first of all, the occasion to introduce my two cohorts from the Programming Language Committee to you. First, our secretary, who gets out a lot of minutes. Quite a bit of volume after each meeting. Miss Mable Vickers from the National Bureau of Standards. And the Vice Chairman of the Programming Language Committee, Mr. Jerry McKinsey, who almost single-handedly edited and got the journal ready for publication.

As Jack pointed out, the purpose of the questions that were written down on each of sessions was to provoke some kind of discussion from the audience. We determined at lunch that you are probably provoked enough at this point, we hope. At least there are no more. Most of the questions that we had written down have been covered and recovered. There are some things that we also felt in discussing the thing just prior to this session, and that is, we're puzzled about what-- we think we know where you want us to go, if we know where you want us to go 10 years from now. It's not exactly clear what everybody wants us to do not exactly tomorrow, but maybe next month or two or three months from now. If the Programming Language Committee were faced with possibility of including a management, information, or database system in COBOL this year, should we do it, not knowing what the data description language is going to be? I'm not so sure now after two days of this. We've made some mistakes in the past, and we don't want to particularly compound these or make others. So, I'll open the discussion to you. Tell us about some of the mistakes. What should we do next year?

MALE:

COBOL, because a lot of them think that's what we're presenting for the last five.

KERRS:

We're doing everything properly? Yes? Positively? Wow. They're rebelling. Shades of Florida. Yes?

MALE:

I missed the luncheon, and I'm interested in this first question. Was that answered during the luncheon? *COBOL Journal of Development.*

KERRS:

What we meant by that, really, was the specification as we've laid it out right now. Is this a sufficient base for us to start in the future? I think we've been talking about that. Maybe we didn't mean to, but for the last day or two. Some people expressed, at least to my estimation, that certain things should be done to the specification, and it does serve as a base to do certain things. There are others, perhaps, that might affect it, but we don't know about, like the data descriptive language. Whatever we come up with. Exactly how that will affect COBOL. Or whether the present specification should interface with the information management system at every level, I should say. I would conclude at least that it does not serve as a sufficient base to do that. Yes, sir?

WRIGHT:

Steve Wright, ADR. What I'm interested in mainly, as far as this is concerned, is not the current *Journal of Development*, but something to replace CIB. Something informal that could be distributed that would tell people what the committees are working on, and some medium through which they could, via letters through the editors or through some other means, participate in the activities of the committees. I understand there is a financial problem in this respect. But I would like to see something to replace the COBOL Information Bulletin, or to extend it.

KERRS:

Yes, sir?

BUSHED:

Anheim Bushed from MIT. I have a suggestion about this. Like the *Communications* by the ACM is read by a large number. This could be one of the channels through which you can publicize the work that's being done in the Programming Language Committee.

KERRS:

Will you identify yourselves, please, as you talk?

SWEENEY:

Jim Sweeney from UNIVAC. I think, unfortunately, it's still not quite clear, the relationship between the CIB, the users, and the *Journal of Development*, and the USASI COBOL specifications, and maybe we could go another 30 seconds on that subject. I think one very valid point that Steve brings up is the general COBOL community's involvement in the development activities of CODASYL. Perhaps we should clarify that at the present time, this is done through membership on the PLC. I think what Steve reads is the fact that other than direct participation in PLC, there is very little communication from PLC to the general COBOL community. I think that's the point that Steve is trying to get some reaction to. Is there any way in which PLC could be improved in its communication?

KERRS:

Yes, sir.

MANNER:

Jim Manner from ??? Data. My own feeling on the thing is, if it seems to be a financial problem to get this information out, I'm sure that the community or those interested would be willing to subscribe to bear the burden of the publications to get a direct communication line to the PLC. Keep up to date with what you're doing. Make it an informal type of presentation so we're not too far behind.

JONES:

I think we've gotten the message very clearly that people are interested in having a much better form of informal and more frequent update as to what's going on. I think all I can say is that the Executive Committee will proceed to try and work out some arrangement whereby this can be done. I'd prefer not to get into the situation where we've got to sell this for a quarter a copy, or something like that. Perhaps some of us would be willing to publish it occasionally and take turns at it, or some arrangement like that. But we certainly will explore that. That's one of the pieces of guidance that's come through very clear, is that we haven't done a good job of that. Mary?

HOLLIS:

Mary Hollis, ISL. Actually, if we could expand it a little bit beyond the specification and get some of the results of implementation from time to time. I think it would be very helpful to those who are using it. In other words, in the CODASYL area, we are essentially concerned primarily with specification of the ???, right? Whereas it would be very helpful if some means were available to say something about the implementation on top of the specifications.

JONES:

In what vein? You're certainly not suggesting that we should write a letter and say NCR's COBOL compiler for the Century 100 is no good. You're not talking about that kind of implementation comment, are you?

HOLLIS:

Well, no, because I think that if you wanted to, I guess to support that, like you would have a very raw analysis.

JONES:

We'd have to have a very good lawyer. [Laughter] We'd have to have a very good lawyer and a lot of money is what we'd have to have. But seriously, I was interested in what vein you were asking for implementation information.

HOLLIS:

In general, we find, for example, that certain implementations are very difficult to handle and are very time-consuming. And there are ways around them. In other words, user hints and not saying that the implementation is all sour or anything like this, but I think some of this would be very beneficial. You would expect certain things to be in a COBOL compiler closer to specifications. It just isn't necessarily so. It would be good to have a forum of some kind for this, other than just is the specification is relevant.

JONES:

That's very tough for us to gather and present that information. Steve?

WRIGHT:

The thing I want do is I want to present half-baked ideas, and I can't do this through the *ACM Journal* because I know that if I don't put something in there, somebody's going to attack me next month in the letters to the editor. But there should be some place where you can present the work in paper that you know half of it is warm. I want a forum that I can present this kind of thing in because this is most useful. Finished papers are a year behind the state of the art.

JONES:

Jean?

SAMMET:

There are at least two mechanisms that I suggest to you. One which many people are familiar with is the monthly put out by the ACM Special Interest Group on Programming Languages, SIGPLAN. SIGPLAN Notices, and a number of half-baked papers have come out there. I don't mean this in a derogatory sense. [Laughter]

JONES:

Yeah, that's a complimentary half-baked way. [Laughter]

SAMMET:

Yeah. What I do mean-- Well, there's another one, namely what was the newsletter of the Special Interest Group on Business Data Processing, now renamed *DATABASE* with a slip-through cover, but would still be ability for informal papers. SIGPLAN Notices comes out monthly; *DATABASE* I think is only going to be quarterly. Am I right? So there is a time factor. But all of these are informal-- I don't even like to use the word "publications" because then the editorial people get very upset. These are informal news bulletins with the express purpose of allowing this kind of informal documentation, which is not refereed, which is merely looked at and edited to make sure that it at least makes some kind of sense, and in which people can correspond back and forth. A number of these things have appeared in those two, and again, other similar issues about these special interest groups.

JONES:

I do think that CODASYL should provide the opening for you to-- any such ideas which relate specifically to us. You certainly should have a way of presenting this stuff to us. I don't think there is any question about that. Dan?

FOGAL:

Dan Fogal. I'd like to just carry Jean's point a little bit further and suggest that the ALGOL Bulletin within the SIGPLAN Notices is really a pretty decent model for this kind of thing, because it has draft working papers; it has minority reports from committees; it has official reports from committees; it has just plain letters; it has arguments about how something should be implemented; questions about what things mean; and little nasty correspondence between individuals, too, if you're really looking for a little love interest along the way. I think the important thing is the ALGOL Bulletin has its own mailing list within the ALGOL Committee and community, and it is also dropped into SIGPLAN, so it gets a wider audience. I think that's the other important thing; it is not necessary to have the official method of sending them.

JONES:

No.

SAMMET:

I think that putting a COBOL bulletin-- aside from certain economic problems which are pervading the entire country. But I think the principle of having a COBOL bulletin associated with SIGPLAN Notices is a perfectly viable one. Well, I don't want to commit those people right now. I think the probability is about 99% that that will be quite acceptable, aside from any economic problems that might exist.

JONES:

As I say, we certainly will explore that method and other methods of getting this information to you. Certainly, I would hope that through the vehicle of this meeting to some extent, you at least know the names of people to write to. We would intend to make this well-known as to how to specifically correspond the informal corporate animal known as CODASYL. Do you have any other questions or comments you'd like to make?

I would like to raise a question to see what sort of a response might come. As we were told yesterday, the X3 activity, I guess it is, has formed a group to work, say, specifically on data descriptive languages. It seems that I've heard quite a bit of discussion here, which indicates that there is at least some interest in CODASYL working on such languages. I would like to open the question of, what does anybody think the relationship should be here? In other words, do you think it's desirable or feasible or impractical or what for CODASYL to say, go off and try to develop a data descriptive language and what might turn out to become a petition with X3. Where would you like to see this work take place? Some people have commented to me CODASYL ought to fold up and go out of business. Other people have commented, "Gosh, there should be no development work done in the organization it's going to standardize. That way, you lose your way check and balance. The development ought to be done in one place, and the standardization in another place." I would like to hear some comments from you folks as to how you feel about this. Where would you like to see this work done? Not necessarily just CODASYL; ACM, JUG, IFIPS. Seems to me there are a lot of possibilities that can be explored. But the message seems to be that somewhere, some development work should be done on data descriptive languages, for example. Where do you feel that should be done? Yes, Brian?

REYNOLDS:

Brian Reynolds, Traverse. I wonder, does it really make any difference? I think it has to be done. If people get together and they feel that they can accomplish something here, let that be the place. I'm sure it's going to get done.

I also have another thing that I would like to bring up, and I think it is an area that CODASYL probably could be the driving force. I think all the questions and the problems that were brought up in the past two days are real problems. I'm not sure that it's the big problem of the '70s. The way things look now is that we can handle programming languages. The industry would know how to use them and how to develop them; keep them coming along. The problem seems to be now is our ability to handle problems in industry. We're getting tied up in the analysis work. I think we have to start developing tools for the analysts to get the job out. I think that much of that could be done in this area. Some of the things that I think fall in mind with some of these where we've been talking in the past couple of days. I think that this data definition is one. On every project it seemed that all around us go through an item analysis of every item in the business. It all goes through the same thing. If we could do it once, get around it, but then have an ability to extract from it one meaning for everything so that our report, when it all comes out, reads the same.

JONES:

You're talking about a catalog of standard data elements type of thing?

REYNOLDS:

Right. I think another thing is that we've implemented all the simple systems in our company. Now we have to get in and implement the difficult systems. I can remember a time of going out and getting the analysis, and by the time I got back to my desk I had the problem pretty well figured out. I think that day is long gone. We have many fortunes and modules. You know, you could say you could make it modular, but when you're dealing with a couple of thousand modules, who's going to recover them all? And who is going to piece them together? I think what we have to do is give a tool to the analyst so he can go out and pick up a piece of information, catalog this, and find out what makes this company ticks. That's the business we're in now, is find out what really makes the company tick. Get these pieces, put them someplace, and run through a SIT process to see what falls out, what should be put in this one, you know, what should be brought together, what should be collected together over here. The way we're going is everything goes into committee now and never seems to come out. Okay, we scheduled programs for six months down the line. It'll be ready for this system. Two years later, they're still later. I think these are the problems that we have to look towards and solve in the '70s. I think our programming languages are there. We can handle it. We don't need something to get the programming languages moving. I think some of the things that the database today-- The reason that we're talking about this is because the hardware has come about, and the hardware is what gave the impetus there. We need something to get us the impetus moving and the analysis there.

JONES:

Some tool to help with sorting out logic that's redundant, logic that's improper or ridiculous in some way?

REYNOLDS:

And how to best get this in the computer. We may have already missed a macro that we can use here. There may be a nonprocedural language that we can use. I think one of the most difficult things is saying, "We have this generalized system here and these requirements here. Does it match?" Okay, it doesn't match. Now, where can we cut and pick? We have to get these tools and develop these things. This is what the system does, and here's where we can fit in our hard coding.

JONES:

Thank you very much. While we may not care where such development work would get done, let me put the question in a different way, then. What would your reaction be if X3 had an activity (I'm only picking on them because they already have identified one) and CODASYL had an activity, both aimed at describing data descriptive languages. Then both organizations came to you and said, "Hey, how about helping us?" Where would you feel your support could best be applied? I think that gets down to the question of, who you do think should be doing this kind of work? Yes?

BUSHED:

Anheim Bushed from MIT. I sort of think that X3 is the proper place for standardization, as to really specifying what the standard should be because that's their role. What CODASYL could be doing and probably be doing is developing the job work, or some of the development of work especially related to the business application or data system application. Because data description is quite a widening, going over all kinds of languages or information procedures, as Bob Beemer said, which includes PL/I, FORTRAN, and other sorts of things. I think CODASYL is not a position to make the data description standard, which is the function of X3, but it is in a position to take the data description facility for the kind of problem that the members of CODASYL invested in, and then give very important feedback to X3.

JONES:

Thank you. I might point out that at the time the standardization activity got started, CODASYL very carefully and specifically drew the line and said, "All right." In fact, some of you may recall we had established a compact COBOL, or something like that, at that point. We actually eliminated that specification because we said, "Well, all right, our business in this is development and interpretation and extension, and X3 is standardization." Now, to be right open about it, CODASYL has not made any moves towards trying to do standardization, but X3 has started to move towards development. Without saying whether or not that's good or bad, because I think that's one of the questions we're here to talk about, I think it is a very real question. Should the development gravitate towards X3 so it's a development and standardization redundancy in place? There are pros and cons, I think. I interpreted your remark to say that you feel like the X3 people should do the standardization work certainly, but the development may well be done better within CODASYL. Since I paraphrased your remark, I'll give you a chance to answer.

BUSHED:

What I was saying is the development of work for this data descriptive language comes from several bases, and the CODASYL base is one of them. For certain types of applications, there are other bases. Something like the ACM SIGPLAN, the ??? description. There are developments on the theoretical background for the standardization. I do think the development work belongs right there. What X3 is trying to do is really not doing the development work. Development work is done by various other people. From this X3 meetings that came out of that committee, what was thought the ACM SIG ??? was one, especially those individuals who have been in those meetings. This is one of the things that-- From all these bases, the Standards Committee, the X3, brings out the standard in, I think the work should be done in CODASYL [inaudible].

JONES:

Okay, thank you. Herb Beta [?]?

BETA:

Herb Beta, Lockheed. First, as a large interested user, I would hate to see the thing-- Further developments get delayed the way I think they would if we waited on X3, so I think, like Chuck said this morning, CODASYL is the horse I'd like to bet my money on. If, subsequently X3 wants to sprinkle holy water on it later, so be it, but I think we ought to go through with it. I'm impressed with the comments Warren had yesterday on this Planning Committee, where it seems like, through all the membership, there are power bases, let me call it, that are expanding and can be expanded further. We can't become an effective voice for a large community. Incidentally, another thought that's crossed through my mind is that it seems to me that now at this time, we are providing a considerable service to the manufacturers of computers. All these problems I hear about of the administrative cross-publications and what have you, I wonder if BEMA might not be the proper vehicle for handling the administrative chore to support CODASYL. I think we could still maintain our arm's-length relationship between the users and the industry

JONES:

Thank you, Herb. Steve, did you have a comment? Steve Wright?

WRIGHT:

Yes. Something that Brian said struck a match in my mind. I see three levels here, really. I'd like to see some people-- He mentioned a two-part analyst. I'd like to see some people trying to do this and break their necks at the individual and user level. This is the first level, when somebody goes up and tries to solve the problem that he sees first. Now, at the second level, CODASYL comes in, and this is an ??? to my problem now, and then they try to propose a solution. I don't care if this is a standard or not. I can care less. If this is useful to me, that's fine. I don't care if you can make this an American standard or not. Now the third level, then, after this has become accepted in the industry and after CODASYL has put its stamp of approval on it, then X3 steps in and standardizes it. I think there are three levels here. First, the individual will see a problem and try to solve it themselves. At the second level, we will recognize this as a widespread problem and try to put any solution—not *the* standard solution, but a solution—that is acceptable to the wide segment. And then the third level comes along and it says, "Yes, we're going to make this a standard."

JONES:

That was very, very concisely put. Thank you. Dan?

FOGAL:

Dan Fogal. I'd like to try to respond directly to the question, which is, would you support CODASYL or would you support X3? Both my corporation and my users group, I think, are a little bit weary of anything that smells like a government standard. The more official it gets, the less eager we are to support it. I think this development thing, I'm not sure that there is such a thing as official development. So I personally feel very strongly that development and the definition of the standard have to be separated and it should not place in the same organization. If that's true, then I think it's very clear which one belongs where. So therefore, in response to the question, if we're going to support something, I think there is no question here. As long as Ralph Nader is not a member, we'll support CODASYL. [Laughter]

JONES:

Ralph Nader, you have a comment…? [Laughter] Chuck Greenberger.

GREENBERGER:

Jack, in direct response to your question, I would have to say that I'd have to see who was on the various groups, what their specific task orientation is, and how likely we thought the implementation for their development would be before we would support either one. I think the existence of two groups makes the prospects of either one somewhat dimmer in terms of getting support. Now, there's something like a 50/50 chance to begin with. As you came out, you changed the odds somewhat. But if were to see X3 having what I consider to be the right kind of people and the right kind of objective and the right kind of resources, then I see now reason not to support. Vice versa with CODASYL, or Guide and SHARE, or IFIPs. It depends. Somebody has to do something right.

JONES:

Thank you. Howard Bromberg.

BROMBERG:

It seems that maybe there's a confusion concerning X3's role. Let me attempt to clarify their role. Standardization activity does not involve more, by and large. It only does development work when they recognize a need to have some development work done. No one else is doing that development work. Now, in the case of the question that you have posed, suppose indeed CODASYL decided to take up this goal. Probably a negotiation could be made real easily, and X3 would be happy to give up their development work in that very area. But that's not really a major question. The major question is, should either X3 or CODASYL do the work? I noticed that Herb and Chuck and Dan are representing very large users, and I wonder whether these very large users have ever considered giving a contract to X3 or to CODASYL to do anything for them. I think the answer would be no.

FOGAL:

No contract with a committee.

BROMBERG:

That's right. If you're not willing to spend your stockholders' money for a committee activity, I don't see why we should be in the same boat, because we're spending our stockholders' money indirectly by supporting this committee function. I propose that what we do is we convert CODASYL from a working organization to a management and control organization and have CODASYL look around—and I'm sure they're quite capable of doing it—find money, and use that money for the right purposes of getting company people, as Chuck said this morning, "full-timers" working on these various jobs.

JONES:

Thank you. Any other comments on this? Bill Ollie?

OLLIE:

As an active CODASYL part-timer, I'd like to speak against this proposal that this work only be done by full-time people. This proposal made by Chuck this morning and Howard Bromberg just now. I think that you have to be able to-- If you're just doing development in a committee environment without doing anything back on the base, then you're working in a vacuum. I think you need to be able to do both jobs. I think we've had a lot of success on the Systems Committee. People, I would say, on average dedicate 10% to 15% of their time to a Systems Committee effort. I hope the figure is something like that in my own case, I suspect it is in some of the others. I think this is the way you've got to go. I think this is really the success that CODASYL has had. If CODASYL were set itself up as a legal entity and to recruit people on that brutally competitive job market—believe me, it is brutally competitive—I think it would just confuse the situation, and I'm very much opposed to it. I think with the way CODASYL is function now, as a voluntary thing, from RCA's point of view, we're very pleased to CODASYL. We always have been, as I understand it. I think this is really the way we have to kind of go.

JONES:

Dan?

FOGAL:

I'd say I agree with Howard's statement of the situation. I do not agree with the recommended solution in this point. I think the development that I think is needed is not a development of specific tools and not a development of specific implementations. What I'm looking for is the development of a consensus about the concepts; some sort of a statement as to what we're doing. I was discussing this with Bill and a couple of people. If CODASYL can provide some leadership in developing a consensus and figure out how this should work, there are many, many of us who are quite capable then of going and contracting and hiring full-time people, of spending money to build the tools. Once we have a reasonable assurance that the conceptual structure is valid and that we will, in fact, get a return on that investment-- and I think this is the key. I can get my controller to support the expenditure of a hell of a lot more money to implement something that the concepts of are defined and we have a reasonable expectation will work, then I can get him to let me spend money on one of my own half-baked ideas. So the idea of a consensus and getting some of the brains is, I think, a very valid thing for a committee. As far as implementing it goes, once we have an idea of what to implement, then I don't think there's any bar of implementation. We've all got plenty of money to spend if we knew what to spend it on.

JONES:

Controllers are all like that. Bill Ollie.

OLLIE:

I'd like to go back on that point as well. I think there's a type of person in the computer community that I call a "concept peddler". There are no end of them. They'll go around and they'll peddle new concepts and say, "Look, I've got some great concepts. If you'd like to pay, I'll see if I can implement them." In the industry, our profession, I'm sorry, is just too immature. Strictly not cleaned up, I'm afraid. In software development, you have concept, definition, implementation. I think there's a really hard step from concept to definition. If you can get your concepts defined, then that's the real battle. Just talking about the concepts is really easy enough to gain, and I don't think you have to just stop. I would hate to see CODASYL become the official concept-peddling playground for the computing community, but I don't really think that's what I think.

JONES:

Chuck Greenberger.

GREENBERGER:

I'd like to rise in defense of full-timeness once again. I agree that conceptually you cannot develop an ivory tower group that is concerned solely with conceptualizing and not in contact with how things are implemented and with how things are used. But full-timeness does not imply that you have this kind of ivory tower. You can have full-time people ???, there are a variety of ways. You can have traveling fellows, all MBS or something, where people come in and work six months, sponsored by their firms, who are experienced implementers and users. You can also have full-time guys test your concepts a little bit more thoroughly than they are done in committee. I don't know how the committees work these days. Maybe part-time is a little more of a full-time than it was in my time. But many things were kind of rushed through, and that thoroughly tested it because people were part-time. We didn't write many programs in draft COBOL, used the verbs, and see how it would work out. Even just to write the code down. And that is really necessary when you are talking about ???. I don't know what everyone did when they tested these things. I'm sure the manufacturers' representatives had more time. But the users, speaking for me, I didn't have the time. We also didn't go, to my knowledge, do too much work looking at some kernel implementation with some of these features. Maybe people were thinking about. I think a lot of people thought very fast. If something came up in committee, they churned it and said, "How would I implemented that?" we kind of kicked it around, and then somebody got a better idea. Now, maybe things have changed and we do a little bit job now. But I think the big flaw of part-timeness is the pace. I think the pace has to be speeded up, and events have to be a little bit more compressed because they're shooting at a moving target. If it moves faster than you're going to possibly move, you'll never get anything.

JONES:

Thank you. Jean Sammet.

SAMMET:

I'll just rise to the very last point there to say, unfortunately, there are some things that take a certain amount of calendar time almost independently of how much fiscal time you may spend with them. Ideas take a while to percolate. If it takes a person two weeks to think through a problem, in fact it may take him two weeks; independently, he'll really be spending 20 hours, 80 hours, or 190 hours in doing this. Therefore—this is an argument against full-timeness—it isn't clear that if you're spending full time on this, that you will necessarily proceed faster. You may in some cases, but it's not automatically obviously.

JONES:

Okay. Bill Keating.

KEATING:

Bill Keating, NCR. I've been listening to these arguments, and they're both good, but I'm not sure if they engage on them. For instance, I think you can get the benefit. I realized what it was, it was all shop at home through a committee such as we have now. We had a pool of full-time staffers. Now we follow through on their ideas. Do the checking of ???, do the writing and the defining. Most of the concepts need to be definition staged. I think the pace will pick up. I think it's really the step we reconsider. In fact, going a little further, you might have a full staff to implement it, and it might be implementing on, say, second-generation machines, so none of the different individual manufacturers would get all happy about it. This way you can test the ideas and bring them a lot further a lot quicker.

JONES:

Thank you. Steve Wright.

WRIGHT:

That's the point. I think that if we tried to do that what you'd find is that you kept second graders on your staff and you kept prima donnas on the Definition Committee. I don't think-- No, I'm not sure which way I would go as to these two, but as far as your suggestion is concerned, I think that's where you would find there, that you could not get a good step on that base because the staff was really backing up the committee.

JONES:

Okay. Peg Harper.

HARPER:

Peg Harper, ARBOC. I wonder if the existence of independent software houses shouldn't have more influence on how the CODASYL Committee might get formed in the future than 10 years ago. I know when I worked with Grace Hopper, I never had to worry what this hour was being charged for because there was this great company somewhere that managed to pay my salary. But since I work for ARBOC every hour of every day is chargeable to something. I think that there a lot of us maybe working for independent software houses who could contribute if there was some way we could handle the finance situation somehow.

JONES:

Yes?

WICKHAM:

Bob Wickham with ADR. The answer is no.

JONES:

Herb Beta.

BETA:

I'd just like to have Howard and Chuck explain-- I'm not sure as to what their proposal way in regard to the composition of this whole kind of stance we're talking about. I'd like each of them to say what they have in mind.

JONES:

By the way, I'm going to shift gears onto another subject here pretty quick because I think we've beat this quite a while, but please give a quick answer.

BROMBERG:

The strengths of CODASYL are two-fold: One, it provides a platform for near strangers and competitors to meet and come to a sort of happy agreements all around. The other is that the federal government, for some strange reason, was a member of CODASYL. CODASYL, one day, decided that they'll have [inaudible], and it will be kind of a standard because they'll really take advantage of their strengths. What's the single weakness of CODASYL? It took 10 years to for some lovely ??? information algebra, E tab X, and COBOL. When we calculate the cost of this ten-year development, I'm sure it'll be absolutely astronomical. Now I'm suggesting that we use the strengths to attack the weaknesses. What we do is we have same competitors and strangers get together and decide the detail of what they write—namely writing and RFP, the request for proposal for this, putting this thing up and seeing which of the smart guys, you know, the Peg Harper companies, the universities, would like to respond to this proposal, and then get all of the users to fund the same thing. Instead of 10 years, we'll get it done in one year.

JONES:

Chuck Greenberger.

GREENBERGER:

[Inaudible] Steve Wright doesn't think it's a good idea. I think if there were money available, say, to have a full-time nuclear staff working with part-timers, and the nuclear staff could be employed-- it depends on who they get to raise the money, if that's an important case. But obviously, it probably should be CODASYL that pays them, or some such successor organization that has to get the money somehow. When you think about BEMA dues-- I thought about BEMA dues. Everyone shakes their on when they're manufacturer side. But in effect, this is the tax on every computer. It could be assessed on basic revenue, for example, which would be in effect a user tax. The virtue would be that it wouldn't be collected by the government; it would be collected like a sales tax on users.

JONES:

The markup is maybe higher that way than any other.

GREENBERGER:

I don't know about BEMA, but they seem to be an organization-- I don't know enough about the inner workings. If they work properly, this could be feasible. The problem there that bothered me, why I didn't talk about was that there's no effective check and balance on this money. If BEMA and say CODASYL were to merge, there would be no way for the users, who are ultimately paying this rate, to pass on how the money was spent. In the federal government, generally if you don't like how the government is spending its money, you throw the rascals out four years later. Now, maybe we can throw the CODASYL Executive Committee out by a vote, and the BEMA management could get to work.

JONES:

Well, I always keep a resignation right here. [Laughter]

SAMMET:

Jack, if I might make a purely personal comment. I think the greatest disadvantage, from an organizational point of view to CODASYL is that it's self-created, self-appointed, and in fact, there is no check and balance on the CODASYL Executive Committee. I'm not saying that the Executive Committee is bad and all the rest of us ought to be thrown out--

JONES:

We haven't gotten in the way with any money, anyway, because there has been no money.

SAMMET:

But I pose this very seriously (and again, this is purely personal opinion), if any serious intention is to be given to various schemes and having CODASYL evoke or involve and get money, then I think some mechanism has to be made for elections, for the definition of an organization, and so forth and so on.

JONES:

Oh, I think you're absolutely right. I think we'd have to become much more formal. When you start fingering the dollars, then you've got to be much more formal. Okay, one more comment. I'd like to switch gears here, however, onto another subject.

FEMALE:

May I ask a question. Don't we have a mechanism in the National Bureau of Standards to set up an organization where we bring in the various expert teams from industry for a full-time basis? Wasn't this set up so that we could do this? The mechanism is already there.

JONES:

We already staggered down that potholed road, I believe, trying to work-- Didn't we explore with the Bureau of Standards the possibility of doing something like this?

FEMALE:

But I think the mechanism is there.

JONES:

It's tough to get fellows in that scheme, because the people you get usually wind up being given such an assignment for training purposes as opposed to contributions. In other words, the good guy, he's kept home to contribute. This is a little bit of a problem.
A question that I've been quite interested in, because I really don't much feel for it, is a question of JCL languages. Now, it may be that nobody else has had any trouble with these. [Laughter] I'd be willing to admit that everybody's smarter. But the subject was brought up, was mentioned as to is it feasible to talk about a common job control language? Is it timely? Is it something that should be pursued? Would it be reasonable for CODASYL to pursue such a language? Say it loud enough so we can record it, Jean.

SAMMET:

As a sure point of fact, there is a committee under X3—namely, X3.42F—which has exactly that charter, namely, to investigate the feasibility on a common control language for operating systems, which is meant to include time-sharing systems. That's a fact.

MALE:

When did they publish them?

SAMMET:

They've only had two meetings. It's a relatively new committee.

JONES:

Would there be the feeling then that this is an area that CODASYL should not try to explore in any way? I'm trying to draw from you a feel of what kinds of things you would like for us to work on. In the case of data descriptive languages, X3 does have a committee, but there was some feeling expressed that CODASYL should get on and look at this, too. Or maybe I'm the only one that got that impression, but let's say that's true. Is the same thing true with job control languages, or not? In the back of the room. Tax?

METAXIDES:

I thought the implication, at least what we've said, is that we ought to keep working on this problem for a couple of years. [Inaudible] we have been specifying the data [inaudible].

JONES:

No, I changed gears to job control languages.

METAXIDES:

But I just [inaudible] data description languages.

JONES:

Oh.

METAXIDES:

Maybe you picked up on the [inaudible] JCL.

JONES:

We tried to make a distinction between data management languages and data description languages. There's been a lot of talk about splitting a data descriptive by developing or designing a data descriptive language, which is free and independent of COBOL, and, in fact, hopefully, all kinds of procedural languages. Procedural language itself. I thought we had raised the point that there is some work going on along this line in X3, and the question was, should CODASYL also pursue this work further? The answer seems to be yes. Then I was trying to bring that same kind of discussion around to job control languages in order to see what the feeling was there. Any comments? Over there.

SCHUBERT:

Dick Schubert, BF Goodrich Chemical. I represent a user, which, perhaps, has led a very sheltered life in the last seven years. In 1962, we elected to use COBOL as the main programming language. Wound up selecting a language called DCOM, which had some very interesting features. It incorporated most of the capabilities of COBOL. It also happened to have the very powerful floating point arithmetic capability like ALGOL. In addition to that, had the mass storage handling facilities. It had a built-in report writer, decision table as a language. Then in recent years, we took it upon ourselves to add communications capabilities to the language. Also, there is the database system language. Now, the discussion of whether we need communication capability in COBOL and the database management facility, and perhaps something that hasn't been mentioned, the decision table, it's not academic but ???. It's something that has enabled me to generate the tremendous amount of productivity among the people that work in our company. It is something that I am going to be very sorry if I have to give up. I look upon going to the next generation or the third-generation computer in order to drop down to a lower-level language, namely COBOL. Mainly because it does not have the facilities, which I feel make it a perfect language. I see all these topics, I think the systems in the CODASYL Committee had better get busy and implement or represent a ??? of the things that you have mentioned here, including the database facility.

While I'm talking, I'll make an observation. Dick Kerrs rather bothered me with his statement that he is terrified of the thought of a change in COBOL, because he may do the wrong thing. In my estimation, doing something that perhaps is wrong is better than not doing anything at all.

[CODASYL Disc 12]

SCHUBERT:

If you don't do something to improve COBOL, it's going to become a dead language, like many of its predecessors. It's the time to move, and move very quickly. I think that the development of PL/I sealed the vacuum that was created by inactivity and the further development of COBOL. You may all argue with me on that point, but I'm just saying. [Applause]

MANATIVE:

Herb Manative [?], IMS. I was going to put my hat on two months when I was Director of Computer Information Systems for an International Paper Company. We too have got the COBOL in 33 computers in 25 locations in the United States and Canada. We also put in a communications package that became available. We wrote our own switcher after two years of running CCAP because we couldn't get anything but B-10, Batman, Superman and all the other things that were IBM [laughter]. We needed a file management system, so we bought Mark IV, and it's doing a damn good job for us at International Paper Company. We had a big profit-planning problem where the marketing organization had reorganized every week they come up with a new product, so we wrote our file management problems to be able to react to forces that senior management put on us that we couldn't anticipate. I'm saying that the reason why the Goodrich's and the International Paper Companies and the thousands of other people who are trying to get a job done for their company are not here is they're sitting at home, doing the things that you people had better get off your asses and come to decide. [Applause]

JONES:

Thank you. Any other comments? [Laughter] Well, I think that was well said. Dick, you want to…? Have any comments?

MALE:

Jack?

JONES:

Yes?

JIM:

Jim ??? from Control Data. I think one thing in listening to some of this is one of the assumptions that I think was made yesterday and I think is till carried through today is that we should not put this proliferation of elements in the language and the fact that we make it very complicated and complex. So maybe we're not giving the users the credit that's due to them, in that I think these people, if anything, are advancing at a faster rate than we are, and we're holding back, thinking that we have to keep it simple for the business types, which I don't think is the case. I think these users are implementing some very sophisticated systems, and we have to try to support them.

JONES:

Thank you. Over here.

McGEE:

Are we changing gears again?

JONES:

No. Say whatever you wanted to say. Just identify yourself, please.

McGEE:

Bill McGee, IBM. I just had a question on one of the questions on the questionnaire. I was wondering if somebody would comment on it and elaborate on it so I can finish filling out the questionnaire. This is toward the end. The question is asking, "By what means should COBOL attempt to achieve independence?" And there are apparently four techniques listed: By more precise language specification; Less precise language specification; Microprogramming; The same as today. I'd be very interested in hearing somebody talk about what is meant by those.

KERRS:

What prompted the question is a problem that we're consistently faced with in specification. It really is a change. It's trying to point up, I think, a change in mood so far as the committee is concerned. In the beginning, most certainly, specifications, which were later considered to be ambiguous, it turned out were strategically ambiguous so that they might be implemented in a machine-independent manner. We have people on the committees who feel that the only way to achieve true machine independence is to specify the hell out of something to the place where you can't do it any other way but this way. So in effect, really, what we're trying to get out there in this question is, is this the right direction that we're taking? I think that's the way we're going. To more and more precise specification is the way we're going so far as the language is concerned. Is this going to be eased by hardware in the next generation? I don't know. Will these machine-independent problems, brought up by trying to achieve machine independence, some of them go away with hardware invention? Or should we backtrack a little bit and make some parts of the specifications deliberately less, perhaps not precise, but not zero in quite so accurately on precisely how to do something as a means of achieving this independence. That's what we meant. I don't know why we didn't say that.

MCGEE:

Was microprogramming mentioned as one possible innovation that would commit greater machine independence?

JONES:

That could possibly ease it, yes. That's all it's meant to be.

KERRS:

I'm at your pleasure for questions. Let me say this: While we had scheduled to be here till 5:00, I certainly have no intention of standing here and trying to think up funny questions just for the sake of not disbanding the meeting. What I want to make sure is that we've covered the subjects that you want to cover and that we want to cover. People that want to say something have had a chance to say it, and when that's all through, I'm through. I'm for going. If you have comments or questions, let's hear them. When you're all through and we're all talked out, we'll give up. Bob?

WICKHAM:

Bob Wickham from ADR. I have made a speech here. Want to listen to it? Or I'll cut it down. I'm here for the first time today. Steve Wright invited me out. I've enjoyed the two days very much. I must say that I feel personally that this is one of the most powerful groups in the country in the computing field. We represent users, manufacturers, software builders—the whole gamut of things. This is good. This makes it powerful. CODASYL has produced a product, COBOL, that I think is a great product. It's a language. It's at a particular level. I think we should raise our sights one level above this. Now, what Hank said this morning about the data managers, and it was reflected by somebody earlier this morning that this is a higher level up. I think that this conference could do itself a really good service by resolving that the Systems Committee should investigate this as a new language that will tie in COBOL. Yes, we're not going to just throw COBOL out. We're not going to throw PL/I out. We're not going to throw FORTRAN out. But we're at a level now, and we can do something if we will get off our tail and do it. Just make a simple resolution, and make it public that this is what we want to do. This is what I'd like to see this conference do.

JONES:

Any comment on that? Yes, Bob Bemer.

BEMER:

Bob Bemer, IBM. I'd like to comment on the amount of time that it currently takes for the specification to be developed and standardized, as well as comment on the competing groups doing the same work; we prompted the question several times in this session. I wonder if a great deal of thought has been given in, perhaps, different groups joining together, such as the three or four DDL groups that are in existence, instead of each group doing the same kinds of work, trying to answer the same questions. In many cases, these different groups are looking at different parts of the subject, and perhaps it's then valid to have separate groups. But I think in a lot of cases, there's an overlap of work. Earlier in the meeting yesterday a question was raised by Bromberg, are we spending twice as many dollars and twice as much time to come up with the same thing? I just raise that question, whether or not there shouldn't be an effort to invite the USASI group to go in the CODASYL effort, that type of thing.

JONES:

Thank you. Bill Ollie.

OLLIE:

I'd like to speak to the gentleman here on the front row. He poses that we do develop a common language on a higher level of COBOL. I'd like to repeat the statement that I made earlier this morning, or yesterday morning, or both—I can't remember. I think this is the goal, and I think we have to keep that goal, but I honestly do feel that today, May 1969, that we still have quite a lot to learn before we're in a position to do a polished job on that language development. I'm saying that because I work for some of the top people on the Systems Committee. I think I work with some of the top people in the computer community on this type of thing. But I do think that here in a couple of years, something like that, it'll be a very meaningful thing.

JONES:

Bob Wickham, you want to answer that?

WICKHAM:

I would like to answer to that in two ways. Number One, you're right, but you're wrong. And 10 years ago, you couldn't reach the polished goal of COBOL as it is today. You couldn't visualize it. Nobody in this room could visualize it, including Grace Hopper, who, I'm afraid, is not here. You reached a very good goal to finally standardize nine years after you started. Now, I'm saying that to pile this on top of COBOL and to restrict this conference activity strictly to COBOL is the wrong way. We want another language. You want a name for it? I picked it up this morning. You pick it up from "data" and "modular." DATAMOD. I'll give you a name. It could change next week, I don't care. But this is going to advance a great deal faster than COBOL did. I like COBOL very much. But we need something above that communicates with COBOL that communicates with PL/I, FORTRAN—everybody else. As Tax put this morning, "I/O independence"—that's what we're looking for. That's all we need. Well, that's not all we need, but that's the thing we see today that we need most.

JONES:

Ed Jones.

JONES:

Ed Jones, Gillette. I think the strength of CODASYL and the COBOL is in the support it has received from users and whatnot. COBOL is the I/O-oriented language. Data manager is an I/O-oriented language or function. There is no reason why this could not be developed by the committee that has the support—CODASYL. If people decide to link it with FORTRAN, they can compile data manager, which is on top of COBOL; link-edit; and you're in business. You've got to function. We no longer have these oddball languages. It's important, I think, that implementation take place always before standardization; not worry about standardization before definition and implementation. 1960, we have the first COBOL compiler. I was doing an installation, and it was there. It was real life. And it saved our neck. In 1963, we pass mass storage. When I had to work with random access on a 360, it was there in COBOL. There may have been some little standardization problems crop up since; there were some implementation. But at least the function was there. I think that maybe data manager could be brought in as part of COBOL, be there, and less worry about standardization and tying it to other programs through link editor.

JONES:

Thank you, Ed. Dan, do you have a comment?

FOGAL:

Yeah, I was just going to say that I disagree with Bill Ollie's comment from this point. I don't quite understand why he seems to feel that it is necessary to produce a product. I guess I'm thinking that as far as CODASYL is concerned, it is the process that is important to me or to us, not necessarily the product. The product will come once the process has begun, but the process, the interaction, the development, the exchange of ideas-- Jean made a point about it. It takes some ideas a while to ripen, and they don't ripen at all until you start them going. I think this exchange, and the process by which this stuff develops, is the thing we thing we need to get going on. Still, there's the products to develop.

JONES:

Thank you. Bill?

OLLIE:

I'd like you to read the other book sometime when you've got two weeks to spare, because we are on the path, and I feel very strongly that we're on the path. I may be influenced very strongly by my period on the Database Task Group. And the time it was taking, and it's still taking them to do a development effort in committee. Now, they have a base of two systems: IDS and general work with APL. This type of development—and I'm not going to tie it to the Systems Committee—but this type of development you're talking about has a base of at least 50. Really, there were so many of these things. That doesn't make it easier. That makes it harder. And we do have to study these things in a lot more detail before we're in a position to do this polished job without a problem there. I'm sorry. I stick by my position.

JONES:

Thank you. Dick? Do you have a comment, Dick Kerrs?

KERRS:

Ed Jones mentioned one of my preserved words. He was talking about implementation before standardization, and he might give me a chance to defend myself a little from the gentleman…

[End of recording]