ACRST-1833

ORIGINAL

OFFICIAL TRANSCRIPT OF PROCEEDINGS

Agency: Nuclear Regulatory Commission
Advisory Committee on Reactor Safeguards

Title: Subcommittee on Computers in Nuclear Power Plant Operations and Subcommittee on Instrumentation and Control Systems Joint Meeting

Docket No.

LOCATION: Bethesda, Maryland

DATE: Wednesday, February 6, 1991 PAGES 1 - 228

ANN RILEY & ASSOCIATES, LTD.
1612 K St. N.W., Suite 300
Washington, D.C. 20006
(202) 293-3950

PUBLIC NOTICE BY THE

UNITED STATES NUCLEAR REGULATORY COMMISSION'S

ADVISORY COMMITTEE ON REACTOR SAFEGUARDS

DATE: February 6, 1991

The contents of this transcript of the proceedings of the United States Nuclear Regulatory Commission's Advisory Committee on Reactor Safeguards, (date) February 6, 1991, as reported herein, are a record of the discussions recorded at the meeting held on the above date.

This transcript has not been reviewed, corrected or edited, and it may contain inaccuracies.

UNITED STATES OF AMERICA

NUCLEAR REGULATORY COMMISSIO.

***

ADVISORY COMMITTEE ON REACTOR SAFEGUARDS

***

SUBCOMMITTEE ON COMPUTERS IN NUCLEAR POWER PLANT OPERATIONS

AND

SUBCOMMITTEE ON INSTRUMENTATION AND CONTROL SYSTEMS

JOINT MEETING

Nuclear Regulatory Commission

7920 Norfolk Avenue

Bethesda, Maryland

Wednesday, February 6, 1991

The above-entitled proceedings commenced at 8:30 o'clock a.m., pursuant to notice, Harold W. Lewis, Chairman, presiding:

PRESENT FOR THE ACRS:

H. Lewis
W. Kerr
I. Catton
P. Shewmon
C. Michelson
J. Carroll
E. Wilkins, Jr.
C. Wylie
P. Davis, Consultant
W. Lipinski, Consultant
T. Rotella, Cognizant ACRS Staff Member
M. El-Zeftawy, Federal Official

PARTICIPANTS:

L. Rib, AECL Technologies
N. Ichiyen, CANDU-3
K. Scarola, ABB/CE
G. Remley, Westinghouse
Barry Simon, GE

P R O C E E D I N G S

[8:30 a.m.]

MR. LEWIS: The meeting will now come to order. This is a joint Subcommittee meeting of the Advisory Committee on Reactor Safeguards-Computers in Nuclear Power Plant Operations -- I didn't know that was the name of our Subcommittee -- and the Instrumentation and Control System Subcommittees.

I'm Hal Lewis, Chairman of the first named Subcommittee and Bill Kerr, to my left, is the Chairman of the Instrumentation and Control Systems Subcommittee.

The ACRS members in attendance are Jay Carroll, Ivan Catton, Carl Michelson, Paul Shewmon, Ernest Wilkins, and Charlie Wylie. Also in attendance are ACRS Consultants Pete Davis and Walt Lipinski. I don't see them. It says on my piece of paper that they're here, but they are, in fact, here in spirit and not in body.

The purpose of this meeting is to discuss computer software applications in future nuclear plants, software reliability assurance, software verification and validation, and software sabotage issues. I might just interject that that's news to me, because I thought we were going to discuss both software and hardware. But that will emerge as we go along.

Tom Rotella, to my right, is the Cognizant ACRS

staff member for this meeting. Medhat El-Zeftawy is the designated Federal official, somewhere in the room. There is to my left.

The rules for participation in today's meeting have been announced as part of the notice of the meeting previously published in the Federal Register on January 23, 1991. Portions of this meeting will be closed due to discussions of company-proprietary information and that has been so noticed.

A transcript of the meeting is being kept and will be made available as stated in the Federal Register Notice. It is requested that each speaker first identify himself or herself and speak with sufficient clarity and volume so that he or she can be readily heard.

We have received no written comments or requests to make oral statements from members of the public. As a general pattern, this is an introductory meeting, so we will go through a number of experiences that people have had trying to cope with the change in technology that has come with the computer evolution in the nuclear business.

We're fortunate today to have some participation from our neighbors to the north, and I'm told that Lewis Rib will introduce the operation.

Do any of the other members want to say anything before we get into the meat of the operations?

MR. MICHELSON: Yes. We did have the hardware discussion yesterday, which is listed in the Subcommittee meeting notice. Today was to be the software.

MR. LEWIS: I see. Ivan?

MR. CATTON: It's my understanding that you really shouldn't separate the two when you're looking to see whether or not the system is going to work reliably. Is there any rationalization for this separation?

MR. LEWIS: None.

MR. CARROLL: That's why we had the meetings one day after the other.

MR. CATTON: So the connection is 12 hours.

MR. MICHELSON: The connection is it takes two days to cover both subjects and somehow you have to have one first and then the next one.

MR. LEWIS: But Ivan's point is well taken. There aren't two subjects. There's one subject.

MR. CATTON: That's right.

MR. LEWIS: I just confess that perhaps I haven't been reading my mail. I didn't know that the hardware was going to be covered yesterday.

MR. KERR: Why don't we decide to do better next time and go ahead with this meeting.

MR. LEWIS: I think we should, but there is an important issue here.

MR. CARROLL: I think you're going to hear a fair amount about hardware today anyway.

MR. SHEWMON: Otherwise it might be nice to have a summary of what was learned yesterday.

MR. LEWIS: Let's do that at an appropriate time and let's not interrupt the speaker. Let's proceed.

MR. RIB: I'm going to give just a very brief introduction to the speaker. My name is Lewis Rib. I am representing AECL Technologies, an American corporate entity, with a local office in Rockville, Maryland.

Among other activities, AECL Technologies is representing AECL, which stands for Atomic Energy of Canada Limited, the AECL's CANDU-3 nuclear power plant design in the United States. The ACRS invited AECL Technologies to participate in this Subcommittee meeting to describe our approach to the utilization of computers in nuclear power plant operations and instrumentation and control systems.

We welcome this opportunity as our second appearance before the ACRS. I would like to introduce Norman Ichiyen of CANDU-3 Design Team, who will make the presentation on the CANDU computer control technology. Normal Ichiyen's background includes a Bachelor's of Engineering from McGill, a Master of Applied Science from the University of Toronto.

He started with AECL in 1973 in the safety systems

concept area. He was the program manager for the computer-based shutdown systems development project in 1980 through 1982. This concept was implemented at the Darlington Nuclear Generating Station. Currently, he is Manager of CANDU-3 Computers and Control Centers Branch.

MR. ICHIYEN: Good morning. As Lewis said, I've been asked to talk about the use of computers and digital systems in CANDU nuclear power plants. As the agenda says, and this is what I assumed you wanted to hear about today, was from the perspective of mainly in the software issues, future applications and plans, issues like software V&V, software reliability, and sabotage was another topic.

How I propose to address these topics is shown in this outline next.

[Slide.]

MR. ICHIYEN: This is going to be awkward. I think I'll stay on this side.

MR. WILKINS: I wonder if you could rotate this about ten degrees clockwise.

MR. ICHIYEN: Is that okay?

MR. WILKINS: Thank you. In order to talk about where we're going to; that is our future applications and plans; I felt it's important that you understand where we're coming from. In CANDU technology, we try to use an evolutionary process. So what I will talk about first and,

again, very briefly is a bit of history of our use of computers in CANDU stations, and then bring you up to date to the latest station that's in service, which is the Darlington Station that just went recently into service.

I'll use that as an example of the state of the technology for our current plants. Getting to future applications, I'll have a bit of a discussion on how we have evolved in our design and concept of digital systems. For this part of the talk, I'll use the CANDU-3 project, which I am associated with, as an example of the kinds of things that we're doing for that design.

As I understand, the main kind of issues you wanted to get at were how to produce reliable software and aspects of it, like verification and validation, software reliability. So rather than talk about these pieces as parts of the puzzle in isolation, I'd rather prefer to talk about our whole software engineering process. We feel that the integrated process, which is integration between development, verification and testing, software reliability, are all the more important part.

In order to be able to discuss this in the short timeframe that I have, I'm going to concentrate on safety-critical software where we've had some experiences with the Darlington shutdown system. In this part of the talk, I'm going to talk about our experience licensing the shutdown

systems on Darlington; more important, what lessons we learned from Darlington or what lessons we hope we've learned from Darlington; and, how we're applying these lessons in what we're doing in the future.

For AECL and Ontario Hydro, we feel that the important movement is in the area of standards, and I'll describe why and some of the aspects of that later. Then I'll move into some of the fundamental principals of a high level standard for safety-critical software which we have just now issued for our own internal use.

When I'm talking about these fundamental principals, they really embody the features of V&V and software reliability. So that's how I propose to get at those issues, while talking about these aspects of the standard. Next I'll talk about the overall status, where we are with this program, where we will be in the future.

As a special item, I was asked by Tom Rotella to talk about our experiences with the Bruce fueling machine incident which I assume most of you are aware of. It's an event that happened about a year ago. I'll describe it very briefly with some conclusions and lessons learned from that experience.

[Slide.]

MR. ICHIYEN: Starting off with our experience with computers in CANDU, this is just a simple slide, just

meant to show the timeline over the years of what we've been doing with computers and, on the vertical axis, the degree of computerization that we've used in these plants.

From our first plant at Douglas Point back in the late 1960s, we had a fair degree of computer control on a number of the main processes. With respect to reactor and process control, this has steadily increased so that at the point where our CANDU-600 designs were in service in the early 1980s, we had pretty well reached full computerization of all systems.

Darlington is, I would say, about 99 percent there, with most systems having computerization in some aspects. Only in the late 1970s did we start in the use of computers in production systems. Back in the late 1970s, we used computers for monitoring of important variables and shutdown system variables at the Bruce A reactor.

In the early 1980s, we used trip comparitors, digital trip comparitors for the shutdown systems, for the process trips. On Darlington, for the shutdown systems, we have full computerization. I'll talk about that. The reason it doesn't show 100 percent computerized is because not all of the other safety systems have full use of computers.

MR. SHEWMON: You'll explain later what you mean by full computerization?

MR. ICHIYEN: Yes.

[Slide.]

MR. ICHIYEN: In terms of historically, again, I said I would talk about Darlington. In answer to your question about computerization, I think this gets at that. Again, I'm trying to compress this into a short period of time. So it is fairly general.

We have three kinds of classes of computers on the Darlington system. In Canada, we've called the main control computers DCCs, stands for Digital Control Computer. We've used that terminology right from our first reactors. On Darlington, all the reactor and process control is done in the DCCs. It's a dual redundant central kind of system with triplicated channels and dual redundancy on the computers.

On Darlington, an additional difference that we didn't have on the CANDU-600 design was that the device logic control was done in PLCs, with the Ontario Hydro proprietary design called the OH-180.

Computers were also used for the operator interface in the main control room, used for alarm annunciation and data logging. In CANDU, we have on-line fueling, I think as most of you are aware, and that is done on computer control. So we have separate fuel handling computers for that function.

In the safety systems on Darlington, that was the

first time we have used full computerization of the shutdown systems. What I mean by full computerization is we have a computer, set of processors that carry out the trip functions, the trip decision functions. If the heat transfer pressure is below a certain level, then it says initiate the trip signal.

We use it also for operator displays, the interface to the operator in the main control room, triplicated channels of information. Both in the main control room and in CANDU designs, we have a secondary control area for the seismic events, as well.

It's also used for operator aided testing. In CANDU, we've used the philosophy that we test the shutdown system right from the transducer to the final elements, and we do that through actually inputting the pressure signals and pressure transducers and checking that the channel, in fact, does trip.

MR. LEWIS: I wonder if I could interject a couple of questions. One is when you use the term computer all through this, do you mean digital computer or are you just using the term computer generically?

MR. ICHIYEN: In all these cases, they're general purpose computers, except for this one which is a PLC.

MR. LEWIS: I understand they're general purpose, but are they digital or analog?

MR. ICHIYEN: Digital, yes.

MR. LEWIS: They're all digital.

MR. ICHIYEN: Yes.

MR. LEWIS: Second question. For example, this last issue of testing of the shutdown system; when you say testing, you mean put in -- you said put in the pressure signals and temperature signal and what have you and see that the system works.

But you don't try all conceivable malfunctions within the system to test it.

MR. ICHIYEN: No. Historically, this is a periodic testing. Again, moving back a bit, in Canada, there's a requirement to show an unavailability that the shutdown meets ten-to-the-minus-three. In order to do that, you have to do periodic testing largely, historically, to detect hardware faults that have occurred since the last time of testing.

MR. LEWIS: I'm just ceasing this example to ask a deeper question. By testing, what you mean is assuring that the system will perform as required if it gets the expected malfunction signals, but not simulating failures in the computer that could generate off-line strange signals. That you don't do. That's V&V, which you will come to, presumably.

MR. ICHIYEN: That's right. We do things like

self-checks and self-tests in the computers to try and get at that aspect of it. Periodic tests are -- for hardware, they're pretty thorough because you actually -- as far as the shutdown system knows, it can't distinguish whether this is an actual challenge from an event or a test.

We don't test just the processor, for example.

MR. LEWIS: I understand.

MR. ICHIYEN: In the past this was manually done and now we have it -- those controls are controlled by a separate computer called the Safety System Monitor Computer. As I mentioned, we did, earlier on Bruce, monitoring important shutdown system variables, through a separate computer again, and that's also done on Darlington.

On Darlington, another safety system that had a degree of computerization was in the emergency core cooling system where we use the CH-180s for discreet logic control. I did have one slide to show you a picture of what the Darlington control room looks like. Actually, I've got to flip backwards.

When I mentioned before the shutdown system interface, those are these 12 CRTs; one for Shutdown System 1, one for Shutdown System 2. You can see some of the others. I think that's another safety system, emergency core cooling, and that doesn't have the digital display. So it's marked contrast from what we do with the shutdown

systems.

All I'm trying to point out here is that we have a fairly high degree of reliance on the computer interface through the CRTs, the annunciation and the data logging.

[Slide.]

MR. ICHIYEN: Moving on from Darlington, as Lewis said, the reactor design that I'm associated with now is the CANDU-3. It's really our next generation CANDU after the Darlington station. I've just listed some of the features here, one of which is a very short construction schedule. That aspect of it is a contributor to the directions that we're going in CNI, and I'll talk about some of those later when I talk about the features of the CANDU-3.

I'll just run through these quickly. Modula. design construction techniques which most of the other competitors are using in order to meet this construction schedule; an interesting feature we call 100-year life, not that all components are going to last 100 years, but our target is to have everything replaceable.

On previous CANDUs, it was not aimed for a rapid fuel channel replacement. On CANDU-3, we are aiming at that target as part of this overall target of a 90-day outage within which we should be able to replace all equipment, including steam generators.

As I said, this is a target. Currently, the fuel

channel replacement takes longer than 90 days. We're talking about four to five months is what we currently see now for a complete retubing, refueling, defueling and refueling and the whole process. That's the current state of the design and we're still trying to get that down to within that target.

[Slide.]

MR. ICHIYEN: I thought I'd talk now, since we're talking CANDU-3, how the digital systems are evolving on CANDU-3 from Darlington. So I'm using Darlington as a reference and I will describe what features on the CANDU-3 are different from it. With respect to control, as I said before, Darlington used a dual redundant hot standby kind of configuration for the control systems.

Now we're moving from this redundant central type system to what I call a true distributed control system architecture. There are two features what I feel are the true distributed control system architecture. One is geographic distribution where the processes are distributed throughout the plant in the areas where they have the applications.

The second feature is what I call closing the loop over the highway. A lot of vendors' products that are called distributed control are really distributed processing and a lot of them don't do the closing of the loop over the

highway. What I mean by that is if an input signal is taken at one location but is needed for a control function in a different processor, then that information, in our concept, is done by sending it over the highway for use at the second process.

The other area where we're evolving from Darlington is in the operator interface, where we have the central system that did control and display. We're splitting that functionality up so that there is a separate what I'll call plant display system which is responsible for the operator interface for presentation of information in the main control room.

On safety systems where we're evolving to is a higher degree of computerization, using more systems. Emergency core cooling, as I said, Darlington used it for the device logic, but not for the displays. Here we'll use all functions using computers. The other area is software practices. We feel that we are evolving in these practices and those are the concepts we're going to use for CANDU-3.

Largely the rest of my talk deals with this aspect, the software practices that we see we are evolving to.

[Slide.]

MR. ICHIYEN: The agenda said that you wanted to talk about V&V and software reliability. The way I said I

would do it is talk about the whole process in general. As I said, I will concentrate on safety-critical software in order to focus on this in the time that I do rather than talk at even higher levels of generality.

What I will describe is our overall approach for producing reliable software. We don't feel that any one component or factor is sufficient and we use an overall approach. I will discuss the part played by V&V, software reliability measures and so on.

[Slide.]

MR. ICHIYEN: Again, in order to say where we're going to, it's important to spend a little time talking about where we come from and our experiences on Darlington are very relevant in the directions that we're taking, especially with respect to safety-critical software.

I think for those of you who are unaware, we have been having a dialogue, I'll say, with our Atomic Energy Control Board from about 1985 to 1990 about the licensing of these shutdown systems and particularly the software. I could spend about a day talking about all the issues in sequence, but that wouldn't serve the purpose here.

I'd like to characterize it with a few features. One was it was an extremely drawn out licensing process. The regulatory group started off with one set of concerns and the issues kept changing over the years. So we would

react and make some changes. This process was a very, very long drawn out affair.

There were a different set of issues that started in 1987 when the Control Board hired a consultant, Dr. Parnas from Queens University. I think a lot of you are probably familiar with his name. He's quite a well known authority in the area of software and reliability of software and his name is associated with the Star Wars designs in the U.S. earlier. He is a very competent and knowledgeable person and the Control Board utilized his services.

In terms of conclusions -- I should first say what actions we took coming out of that experience on the licensing. Again, these are just some of the major highlights. There are a whole lot of actions that we ended up taking, but the characteristic ones are the ones here.

Again, back to the name Dr. Parnas, this is one of the things that he instigated. We what I called back-engineered a software design specification using mathematical notation, which is known as formal methods in the industry.

Previously we had used an English language specification, a functional specification. I think it's fairly well agreed that a lot of errors that do occur start with that English language specification. English is not a

very precise language. It's quite ambiguous. A lot of the errors that are made in any software engineering process either are errors in the functional requirements or errors in interpretation.

Either way, use of a mathematical notation was felt to help understanding and other features, and I'll go into those later in more detail.

MR. LEWIS: Is this a matter of mapping the output to the input using the Bacchus Nuir notation or are we talking about something else?

MR. ICHIYEN: What we did was we took the English language specification and turned it into the mathematical notation.

MR. LEWIS: I'm talking about which mathematical notation was used.

MR. ICHIYEN: In this case, we used Dr. Parnas' particular notations.

MR. LEWIS: His own idiosyncratic, not one that is used by anyone else?

MR. ICHIYEN: Probably that's true, yes. It's not a standard notation.

MR. LEWIS: I see. Thank you.

MR. WILKINS: Is it at least published by him someplace?

MR. ICHIYEN: He's got a number of papers that are

published. He's probably one of the more prolific writers in the field.

MR. LEWIS: I didn't know he used a notation different from the one other computer scientists use. That's news to me.

MR. ICHIYEN: I think in terms of notation, maybe I'm causing some confusion. He uses certain notation to describe constants, variables and what exact notation is used for that probably isn't that critical.

MR. LEWIS: The one that is used by most computer scientists is called the Bacchus Nuir notation. Never mind. We'll go into that later.

MR. ICHIYEN: The second action that was taken began really through Dr. Parnas' involvement. It was to establish a walk-through in order to verify that the code met the formal software design specifications. In theory this was a doable job, but neither he nor others had really worked out the practical aspects of it. So that was one of the things that took a lot of time, was working out how we do this in practice.

It involved creating new techniques that hadn't been used before, creating what we call program function tables from the code and comparing to the mathematical notations at the beginning. It involved establishing techniques and methodologies for doing that and how you

compare them.

The third significant action is a random testing program. I won't go into more here. I'll talk about it later as we go. It's testing the processor. So it's putting inputs in, but it's not testing the whole system as an integrated system.

MR. MICHELSON: Ivan, you need to use your microphone.

MR. ICHIYEN: Sorry?

MR. CATTON: I was being chastised for not speaking into the microphone.

MR. LEWIS: I wonder if I could ask one question. In the specifications, do you distinguish between reliable operation and graceful failure modes when the system fails? That is, do you go to the next level of assuring graceful failure when there are hardware failures?

MR. ICHIYEN: In the Darlington system, this is the shutdown system and not a control function. So what we're interested in in the safety features is that it's fail safe. So wherever there is any doubt, the action is to -- if it's in an undefined stage or something that's not right, we go to the trip state.

MR. LEWIS: What I'm suggesting is, and that's why I was extremely unhappy to learn that we'd separated hardware from software in this meeting, the software has to

be written in such a way as to accommodate hardware failures, in such a way that it leads to a graceful failure of the overall system.

It's the job of the software to do that. Is that within the specs that you laid down on the system?

MR. ICHIYEN: I wasn't that familiar with the details.

MR. LEWIS: All right. Thank you.

MR. ICHIYEN: I could check into it.

MR. SHEWMON: A variant to that, which I will bring up maybe twice, but at least later, has to do with the Rancho Seco failure in which there was a power supply failure, which is a variety of equipment failure that may not have been safety primarily. But how the system copes with something like that will come up also.

MR. ICHIYEN: Power supply failures?

MR. SHEWMON: Yes.

MR. ICHIYEN: To the computers?

MR. SHEWMON: Well, this was a power supply to instrumentation and control in that case and then different systems went in different directions and the operator wasn't sure where they were. So he had a lot of problems.

MR. ICHIYEN: In the CANDU designs, which is probably the same in other designs, the power supplies for the safety systems are separate from the control systems.

We have a complete separation of safety and control. There is no functional or equipment connections in any sort.

We use triplicated channels in the shutdown systems and there's two or three --

MR. SHEWMON: Fine. I'll ask the question when we come to control and how it might interact with safety, then.

[Slide.]

MR. ICHIYEN: The real issue that came out of the Darlington licensing experience, in our minds, was the lack of an accepted definition of the acceptable quality that software has to have in order to be approved by our Atomic Energy Control Board. As I said before, the issues kept changing. There was a lot of subjectivity and we feel that the real cause of that was that it wasn't a real de facto standard or real standard which set the requirements for what is required.

So our objective now is to create a set of standards, procedures and guidelines for software engineering, overall categories of software. Our first task is --

MR. CATTON: How do you define acceptable quality?

MR. WILKINS: Next two lines. It has to be approved by the AECB. Of course, you can ask a different question. How do you assure that it works? That's a different matter.

MR. LEWIS: And still a third matter is how to be sure that when it doesn't work, it doesn't work in a relatively benign way, which is the third matter.

MR. ICHIYEN: As I said, our first task is the creation of this set for safety-critical software. We're starting with that one because after all is said and done and the plant is licensed, the Control Board has said the effort to license the Darlington system required a lot of expertise and individual effort and a lot of subjectivity even after having the formal specifications and the walk-throughs and so on.

So what they're setting as an objective for Ontario Hydro is to redesign the software over a period of time of five to six years so that these problems do not occur again. That's one of the main reasons for creating this set of standards, so that we don't go into these same problems again. We want to get an agreed set of requirements that we can both work towards and then use that as the reference.

MR. CARROLL: The need to create a set of standards in this area suggests that there are not adequate standards already. What is your view of the existing U.S. standards that are in the software area?

MR. ICHIYEN: I haven't read all of the U.S. standards and so on. From what they're telling me, that the

standards are not aimed at the kind of applications or the techniques that we see or our requirements that we see.

There are a lot of good features in the standards. Some are more prescriptive and too low a level of detail. What we're trying to achieve is a higher level standard which, in a lot of ways, is methodology independent. We're trying to specify what the core requirements are in a way that the Atomic Energy Control Board and the utilities and ourselves can agree with, and then we work on our methodologies.

We present these methodologies to the Control Board and say does this meet the requirements as stated in the standard. I'm saying we're trying to make it methodology independent, but you really can't do that in all areas. What we're trying to do is not limit the methodologies unnecessarily.

I think maybe you'll get some of that picture when I talk about some of our main principals to see whether those are embodied in the other standards.

[Slide.]

MR. ICHIYEN: There are really four parts to our framework that we're developing. One is a categorization criteria. I think everybody would agree that software has different categories in terms of what's required and what kind of assurance requirements do you have for it, some of

which has no impact on safety, some of which is very important to safety, like a shutdown system, which is the last line of defense if there is an event.

The difficulty here is how do you quantify the attributes of what constitutes a category, how many categories, what do you do with those categories. It's clear the highest level being safety critical is an easy one to define. It gets a little fuzzier as you go through the lower levels, like control software, monitoring software, non-real time. It gets well beyond the plant software. You get into analysis software and so on.

But we're starting with the safety critical one and working down from there.

MR. CARROLL: This standard would be broader than just control and protection. It would go into --

MR. ICHIYEN: Analysis software.

MR. CARROLL: Analysis kind of software.

MR. ICHIYEN: We see it as a family of standards. For the real time ones, there should be a very close relationship. So we're writing the safety critical one first which, as I said, has been just issued for our own internal use. We are now working on the other categories. Actually, I've already jumped ahead to my next slide.

[Slide.]

MR. ICHIYEN: Again, the four parts of this

framework at the categorization criteria, the high level standard which I've already described being largely aimed at being methodology independent, and then what a lot of people call sub-tier standards, which are a lot of the details and how-tos and specifics and things which are methodology-specific.

Another aspect of what to do with developed software, which is software that you purchase.

MR. KERR: Can you give me an example of a categorization criterion?

MR. ICHIYEN: Pardon?

MR. KERR: Can you give me an example of a categorization criterion?

MR. ICHIYEN: Largely, the definition for safety critical, and I'll try and say this, safety critical software would be software in a system whose failure could lead directly to a significant release of radiation to the public or to the plant operations.

MR. KERR: Thank you.

MR. LEWIS: In this effort, which is more or less a start from the bottom effort, did you bring in consultants from other industries who have computerized themselves over the years, not just Parnas who is a well known person in the business, but, for example, the 767 airplane, Boeing, is a very highly computerized airplane. I would judge that it's

probably a factor of two less complicated than a nuclear power plant, but not a factor of ten.

The telephone company is probably a factor of ten or 100 more complicated. So there's plenty of industrial experience with people who are trying to make very large computer systems work reliably and in a fail safe mode. Did those people get brought into this effort?

MR. ICHIYEN: We did what we thought was a fairly extensive survey of not only the nuclear industry, but, as you say, other industries. There is one distinction, actually. With the way we do our designs, we felt that the functionality of the trip functions that are true safety critical are not very complicated and the order of magnitude is more like ten or 100 less than some of these larger applications.

The number of lines of code in a Darlington t computer is -- some people will say it's 3,000; some people, if you take out the comments, you're down to 700. It's not a lot of lines of code. We feel that for that specific kind of application, a lot of the applications that are used for larger systems are not as precise or don't give as high degree of confidence and the reliability that we feel that the methods for smaller systems can do.

In Canada, there's a telecommunications organization, Bell-Northern, and all the people, as you

said, the telephone industry are in conjunction with Dr. Parnas. So they're doing work with him in that area, as well. So they're using techniques that are similar to a large degree.

Within the nuclear industry, we did a lot of talking to the people in the U.K. Actually, not in the nuclear industry, but in the formal methods area. We've talked to other people, France, the Westinghouse people and what other vendors are doing in their software practices.

Having surveyed all of that, we feel that for our application the technology or the methodology that Dr. Parnas is refining, I'll say, is the one we feel the most comfortable with that will do the job for us. The thing about Dr. Parnas is that he hasn't defined what we think are workable methodologies.

He's been working on a lot of these things for quite a long time. Things like trace specifications he's been talking about for ten or twelve years. But it's only when you get into real situations that you have to make the methodologies work and you extend the boundaries of the knowledge of a system.

So we've been working with these systems. In fact, we haven't defined our methodology yet. Again, I'm jumping ahead, but I'll say it now. We expect by the fall or mid-year of 1991 to have completed our studies on the

methodologies. AECL and Ontario Hydro are working quite closely in this respect.

MR. LEWIS: The reason I started with the 767 example is that the complexity of the 767 is really not all that far from the complexity of a nuclear power plant. Certainly both of them are well below the Star Wars system. Even in the great Star Wars controversy, Parnas was really pretty much a minority of one on that advisory committee.

So he doesn't reflect at least the majority sentiment of American -- forgive me -- of U.S. computer scientists. But please go on. I'm slowing you down and you're running behind.

[Slide.]

MR. ICHIYEN: What I thought I'd do is talk about the high level standard and its features. As I said, this will serve to highlight some of the principals that we feel we are using in the future. The high level standard defines the requirements on the software engineering process, defines the outputs of that process. It defines the requirements to be met by each output. What we try and do is to specify this as measurable as possible, but, as I said earlier, not necessarily to constrain the methodology to produce the output.

[Slide.]

MR. ICHIYEN: In terms of fundamental principals

of this high level standard, I'm going to talk about five items. The first is the use of the documentation and the mathematical notation. We feel the documentation must describe the required behavior of the software using mathematical functions written in a notation that has clearly defined syntax and semantics.

We feel that using this kind of notation, you end up with more complete requirements. Since it's mathematical, you can verify that the complete domain is covered and you can check it. Using mathematical notation, as well, requirements can now be uniquely interpreted. Dr. Parnas uses this example of statements. One requirement could be that you shut off the pumps if the water in the tank is over the setpoint for four seconds. In English, that can be interpreted in any number of ways.

If you're the software developer, you have to say, well, what does he mean; is that the root mean square of the level, the average level over the four seconds, the median of the level, when the minimum level is over for four seconds. If you do it mathematically, there's no ambiguity. You know what the requirement really is.

MR. WILKINS: Of course, you may not know that the requirement is relevant.

MR. ICHIYEN: Yes. But you could tell if it's right or wrong.

MR. WILKINS: No. Whether it's been met.

MR. ICHIYEN: Other people can review it to see whether it's also correct as stated. One of the problems in the English language is that one person interprets it saying that's right, that's what I understand it should be, but that's not what the author meant and it may not be what the software designer actually implemented.

Again, using mathematical notation facilitates use of mathematical verification techniques. That allows the design to be transformed into mathematical functions for comparison to the requirements directly. We've worked with this for quite a long time. Mainly Hydro has been doing a lot of this work.

We've got it to the point we think it's an actually doable task. We're at the point where we're about to start developing tools that will help us with this process.

MR. LEWIS: I hate to be a troublemaker, but I've been counting pages. You have about ten minutes to go in your allotted time.

MR. ICHIYEN: I may drop the Bruce fueling machine thing. It's been published and talked about, I think, to death and maybe you can ask me questions rather than my talking about.

[Slide.]

MR. ICHIYEN: The second fundamental principal is that the outputs from each development process must be reviewed to verify they comply with the requirements specified in the inputs to that process. What I mean by that is if you think of the development process in three stages; the requirements specification, the software design, and then the coding; the verification process is on each of these outputs.

You verify that the outputs comply with the requirements as specified on the inputs. Where you use mathematical functions, you can verify these against the inputs using mathematical verification techniques.

[Slide.]

MR. ICHIYEN: The third principal is on the use of information hiding. I won't go into that in a lot of detail, other than to say that the -- really what it is, in simple terms, is that we try and -- this is for the effort for maintainability. What you try and do is to assess what areas in the code or things are likely to change over the history of operation and you use this as a guide to how you do your software design into your modules and so on.

In addition, you try and design the interfaces to the modules, to reveal as little as possible about the modules' internal workings. This is, again, to help the maintenance aspect.

[Slide.]

MR. ICHIYEN: Along with the verification and the specifications, we feel that in terms of testing you need to do both systematic and random testing. By systematic testing, the normal things that are done in the industry, you characterize it as white box or black box testing. In white box, you understand, you know what the internal workings of the code are and you test based on that and try and look at discontinuities and so on.

Black box, you treat it as a black box where you don't know the workings of it. You check that the outputs match the requirements that are stated in the requirements. Thirdly, we want to add random testing. We call it statistically valid random testing, which is a contentious issue in the software area.

MR. CATTON: Is this a good place to ask the hardware question again?

MR. ICHIYEN: Could you state that one again.

MR. CATTON: There's a school of thought that says when you're dealing with embedded software systems, you have to test the system, which means software and hardware, if you want to come to some meaningful conclusion about its reliability.

I haven't seen you mention anything about hardware yet.

MR. ICHIYEN: At least in the past for Darlington, we do the integration testing which is how you test the system as a whole. It isn't meant to be as exhaustive as the kinds of testing we do here. In the specifications, if you do the specification right, you're specifying what the computer system must do and then you break it into what the software must do.

If you do the verification right, then the exhaustive testing is really on the software. That's where we've been doing in the past.

MR. CATTON: I know that at NASA-Dreiden, they're very interested in taking one of the computers that's tied up with data evaluation and coupling the whole system together and putting -- trying to figure out what would be nice set signals to give it that are a little bit out of sync with what they should be to see how the whole system operates.

They actually have people trying to figure out what would be a good set of inputs to really test the system. This is not nearly as critical as your shutdown system.

MR. ICHIYEN: When you say testing the system, meaning --

MR. CATTON: They have some hardware. Actually it's from a pilot in the aircraft sending information to the

ground, processed by a computer, and there's a human being in the middle. The information is sent back to the airplane and the pilot has to take an action.

They're seriously trying to figure out how to test that system in one piece.

MR. ICHIYEN: Maybe I need to describe the configuration, because it's not a complicated configuration. In fact, we are testing -- I hope I have a slide of it here.

MR. KERR: Ivan, could you restate your question? I don't understand it.

MR. CATTON: I'm not sure I do either. The question has to do with testing the system, the software drive's hardware. Some people feel that if you're going to establish a reliability for this system that includes computer software, you have to look at the whole thing. They even have a name for it now. It's called embedded systems where the software --

MR. KERR: Given that that may be valid, is it impossible to establish standards for software and establish standards for the total system and test them separately?

MR. CATTON: Personally, I don't know, but some people feel that you eventually have to do the test on the whole system.

MR. KERR: I haven't heard him dispute that. But he is not --

MR. CATTON: I don't know if he's going to dispute it or not. I asked.

MR. ICHIYEN: I think you need to understand the configuration of it because it isn't a very complicated configuration. This is actually a fundamental principal we've tried to use on safety systems or safety critical systems. We make it as simple as possible.

[Slide.]

MR. ICHIYEN: These are what I call the safety critical pieces of the boxes, which are the trip decision functions. These are display computers which are used as the interface to the operator. What we do in testing is we take this as a system. We know the inputs to it, we know the outputs. We isolate that. We test that with varying degrees of testing. Software is done in smaller chunks than unit testing, so on, building up to the random testing that I talked about, for example, would test this whole unit in a random way.

Also, we do systematic tests. So I think, in essence, I am saying that we do test this aspect of it. We don't feel we need to test the whole system which is the operator interface, the displays on the monitor system, displays in the control room, to the same degree. We do test them, but to a lower level or higher level, depending on which way you're looking at it.

MR. CATTON: Thank you.

[Slide.]

MR. ICHIYEN: I'll just move quickly to the last one. Another item that we feel is a fundamental principal in our safety critical software design is the use of hazard analysis. Again, I'm pointing this out because we feel that there's not just one technique that has to be used for safety critical software.

Hazard analysis is another tool in our tool chest that we use to have a higher assurance of the quality of the software. A simple definition of hazard analysis is that you identify failure modes that lead to an unsafe action and eliminate them or ensure the failure mode can be detected and the system put into a safe state.

Dr. Nancy Levison, I guess, is the key proponent of this because she thinks she calls software fault tree analysis and I guess we've coined the term hazard analysis, and I think she probably uses the same term. She was hired by Ontario Hydro as a consultant during the Darlington licensing period. This is one of the extra features that she brought to the design process.

Basically what she's doing is using fault tree techniques that are used in hardware, applying them to software, looking for events, failure modes, and then seeing what things in the software have to happen in order that you

can mitigate that event.

In my mind, what it does is it gives you a more robustness to your software design as opposed to just meeting strict functional requirements.

MR. CATTON: I, today or actually last night on the airplane, read the results of the workshop that I've handed out here. They maintain that using the techniques that you would for hardware, which usually means random failure for software, is incorrect. Really what you ought to do is go back and use the approach that there's a possible design error, which is different, because if there's something wrong with the software, it's a human error somewhere, most likely. The answers you get out of it are different depending on the approach.

MR. ICHIYEN: It's identifying what are the others, and those are the key parts of any fault tree analysis. You don't have to identify how those failure modes can occur necessarily. What you try and do is to mitigate those occurrences.

MR. KERR: Ivan, under the theory of statistically valid random testing slides that he has, there is a distinction between at least a definition of reliability for hardware and software.

MR. ICHIYEN: I skipped over those in the interest of time. I don't know if you want me to go back.

MR. CATTON: I don't know if that statement I made was right. I just read it. I was looking for your response.

MR. LEWIS: We'll take a vote later about whether you're right, Ivan. But I am going to be brutal and try to wrap us up by 9:30 so we can keep on schedule.

[Slide.]

MR. ICHIYEN: In terms of overall status, as I said, the safety critical high level standard has just been issued for use internally to AECL and Ontario Hydro. We plan to have the sub-tier standards, procedures and guidelines to be completed by the end of 1991.

The methodologies that we're using, safety critical hardware configurations for systems, we're planning for mid-1991 and that will probably be slipping till probably in the fall. But it's in that order of magnitude in terms of schedule. We're also working on other categories and standards for those categories with really an undefined closure date as yet.

MR. LEWIS: I'm going to thank you and assume that we can skip talking about the Bruce incident in the interest of staying on time. I know we would be very interested in it and I know we would, therefore, spend at least a half-hour on it. The best way to prevent that is at the beginning. Our purpose here today, I hope you understand

from our questioning, is not in any way to provide unwelcome advice to our Canadian friends, but to learn from your experience while we try to advise our American friends.

So I thank you very much for your presentation. It was very informative. I think we should just go on. I'm told that our next speaker is Ken Scarola, is that correct?

MR. SCAROLA: That is correct.

[Slide.]

MR. SCAROLA: Good morning, gentlemen. My name is Ken Scarola. I am the Manager of Advanced Control Complex Engineering at ABB/Combustion Engineering. I'll be talking about software reliability issues for NUPLEX 80-Plus which is the advanced control complex being used by CE for the System 80-Plus ALWR.

I might add that the NUPLEX 80-Plus advanced control complex is also being used for the heavy water reactor, NPR, at this point in time. To address very briefly what I heard about the relationship of hardware and software, I would say that CE would agree 100 percent that these are not separable issues. In fact, I think most of the industry does agree.

You will see, as I present our verification and validation approach, we definitely use V&V as an integrated process on a system basis and then an entire control complex basis. So these are definitely not separable issues.

MR. KERR: Does complex go with control or with engineering in that slide?

MR. SCAROLA: Good point. Probably both. What I'd like to do is address the issues that we believe are the main contributors to software reliability. There are numerous issues. Software reliability does not stand on any one particular issue. It's a building block defense-in-depth approach and I think all things must be considered.

Certainly we talked about hardware reliability yesterday. What you're going to see in my slides is many of those same points are now repeated here. In fact, I do have some slides that I will throw in that may not be in your handouts that I used yesterday and I will be able to get you copies of them.

Basically these are the software reliability issues that I will address. What I would like to do is discuss CE's experience basically in between here before I talk about the software design process. I'm going to rearrange that from what's in your handout. That will help, I think, set the framework for the software design process because it's based on our experience. Those slides are misplaced.

[Slide.]

MR. SCAROLA: The first subject that I would like to talk about is what we call deterministic design. The

most important part of any software-based system is its simplicity and your ability to prove that it works. When we talk about computer-based systems, they range all over the map. When we talk about systems for a 747 or something like that, I would have to maintain that they are significantly more complex than the types of software that we design for nuclear power plant protection systems.

To give you an idea of what we mean by deterministic designs, in our protection systems, the inputs are scanned and processed continuously. There is nothing like what you do in a complex system where you report data changes by exception and then, when the particular data changes, then you process it for that change. That's not the case.

In our protection systems, we look at the data with every cycle and, in our protection systems, we have less than a 50 millisecond cycle. That data is processed all the time. That's regardless of the state change of the data. Now, that can't be done in or it's difficult to do in very large complex computer systems because you cannot get the performance out of the system.

But if you look at a protection system for a nuclear power plant, and specifically CE's system, there are 16 inputs. There is one output, reactor trip. That's not the case for things like DNBR and local power density, but

all of the other trips are, in fact, that simple. It's because of that simplicity that we can run the system on a continuous scan/continuous process basis. So there are no surprises when things change.

Similarly, the outputs are updated on a continuous cycle. The outputs are not simply updated when the process logic says they need to be. The outputs are always updated, which means that in a protection system, 99.999 percent of the time, the output is updated saying don't trip, don't trip, don't trip.

On the one cycle when it doesn't get that update, it trips. That's basically how the system works. Now, all of the programs are run on a continuous basis, meaning there is no multi-tasking as you would see in most large computer systems. The system does not run with interrupts. All the data is processed on a continuous cycle basis.

Another important fact is where we use programmable logic controllers, which is the fundamental basic technology in our protection system, those machines run without branching, which means when they make a decision, they don't go off and do something because of that decision and then come back into the program. They make a decision, they set a flag, the program continues on, and some point later in the program on a continuous basis that flag is recognized and something gets done because of that.

That's the inherent nature of programmable logic controllers. That's not the inherent way most computers run. Most computers run with branching, with sub-routines, with calls, and you have to force them to run in a deterministic nature. In our CPCs, we do that in CE's core protection calculator because that is an inherently not deterministic computer system.

So we have to write the code in a very structured manner that forces it to run in a deterministic somewhat non-branching type of approach. For the simple part of our protection system where we look at analog variables, pressurizer pressure, steam generator levels, we run non-branching.

MR. LEWIS: I'm really a little bit confused by something here, a distinction you're making which appears to be important, but which I don't understand. My computer at home has a combination of software interrupts and hardware interrupts. Hardware interrupts are branching interrupts, in general, which simply tell the hardware to go off and do something else, and to do something else may involve returning to the original program or may not involve returning to the original program, depending on the character of the interrupt.

But it also has software interrupts which simply set a flag and the next scan time around to see if any

interrupts have been activated. It notices whether they're there and switches off to another part of the program which may or may not involve a return to the main program, depending on what it is.

I don't see how that differs from what you're describing.

MR. SCAROLA: It does in really two senses. First of all, in your system, you're saying there are branches where you may not return to the main program.

MR. LEWIS: Sure. That's dependent on how you write the program.

MR. SCAROLA: In our safety computer systems, that is not acceptable. Where we do branches, we always return to the same point in the program. It's what we call single entry/single exit of modules. That inherently makes the software more predictable that it's going to perform the required function when you want it to.

MR. LEWIS: I don't see that because it's predictable either way. It depends on how you write the software.

MR. SCAROLA: Not necessarily. When you branch and the number of branches and the number of nests that may occur in subsequent branches, these are the reasons why software very often goes off and does unpredictable things. It does things that you are not able to anticipate, like get

stuck in a loop somewhere.

MR. LEWIS: The Bruce event, which I didn't allow the previous speaker to describe, was a case in which, as I understand it, where what should have been a jump to a sub-routine was instead written as an absolute jump, and that's what caused the problem. It would have been better if it had been written as a jump to a sub-routine and come back to the main program.

MR. SCAROLA: I think the point that I'm trying to make here is that the structure of the code and the methods that you use in coding are fundamental to the ability to predict the performance of the system. I agree that you can establish predictability in very complex systems. My point is it's more difficult.

MR. LEWIS: In a sense, what you are saying is you've made a decision that non-return branches are inherently safer than return branches. Is that correct?

MR. SCAROLA: No. I'm saying that --

MR. LEWIS: I'm trying to understand what you're saying.

MR. SCAROLA: I'm saying the return branch is when you branch and you return back to the same point in the code.

MR. LEWIS: Yes.

MR. SCAROLA: That is inherently more predictable

performance than if you branch and subsequently branch again and subsequently branch again and may never return to the same point in the code.

MR. LEWIS: So I had it backwards. You've made a decision that only jumps to sub-routines which return to where they started are acceptable in your world.

MR. SCAROLA: Right. What I'm saying is that's the approach that we take for complex calculations, such as DNBR and local power density. For simple things, such as an analog functional trip on low steam generator level, low pressurizer pressure, the code works even more predictable than that. What I have is a slide here that's not in your package.

[Slide.]

MR. SCAROLA: This basically is a mapping of how the software executes in our programmable logic controllers independent of what the system inputs are doing, meaning the software follows this path every time. If you were to map the software execution cycle of a conventional computer, you would see that the mapping is all over the place. It zig-zags, it goes out, it comes back, it goes to many different places.

A programmable logic controller inherently runs in a deterministic cyclical manner. It never changes its execution.

MR. LEWIS: Forgive me for being stupid, but I'm really trying to understand. It is your belief, then, that there is really no case in which it is preferable to leave the main program and never come back to it.

MR. SCAROLA: No. I can't say that, that is not true.

MR. WILKINS: Let me try something.

MR. LEWIS: Go ahead. Try to explain.

MR. WILKINS: I don't operate quite at the level of sophistication that these guys do, but on my computer I've got a go-sub order, and that's okay. After the go-sub order, you return. But go-to is not okay.

MR. LEWIS: I understand that.

MR. WILKINS: That's not what he's saying?

MR. LEWIS: I think that's what he's saying. In fact, that's common belief among computer scientists. In fact, when C was written, they originally didn't want to put the go-to into it. Now they've put it in, but they said it's strongly counter-productive.

I guess I'm asking -- there are, believe it or not, cases, using your analogy, in which go-to really is the right thing to do; that is, if you get a signal that tells you that the reactor has broken in halt, you don't want to come back to the original program.

MR. SCAROLA: I'd like to move on and just point

out that the simplicity of the software execution is important.

MR. LEWIS: That's certainly right. In fact --

MR. SCAROLA: We can accept that and recognizing there are many ways to make the code simple, we have selected one that we think is the simplest approach. There are others. I'd like to leave it at that.

MR. LEWIS: I couldn't agree with you more. In fact, what Ernest was saying, which was that the use of go-to is discouraged, is certainly gospel among modern computer scientists. Good programming practice does not use go-to. It uses modular systems, it uses predictable systems.

If what you're saying is that one should use good software practices, then I have no problem at all.

MR. SCAROLA: Thank you.

[Slide.]

MR. SCAROLA: The second important contributor to software reliability is the use of field-proven executive software. In NUPLEX 80-Plus, all of our software-based systems are composed of off-the-shelf commercial products with extensive field-proven industrial experience. Now, this includes programmable logic controllers, as I mentioned. We do use PC ATs. There are minicomputers, CRT workstations, etcetera.

All of these systems are bought with executive

software, meaning software that has been in use in the field for handling things such as the input/output processing, the arithmetic functions, communication drivers, and failure detection inside the system itself.

This software is what you would call reusable code, a code that has extensive operating experience, thousands of applications. So we attempt to use reusable code as much as we possibly can because we believe that field experience is the best validation source.

[Slide.]

MR. SCAROLA: Now I'd like to talk about CE's experience. CE has been designing software for safety systems since the mid-1970s, basically with the core protection calculator for ANO-2. That was our first experience. Since then we have put CPCs in all of our plants.

In addition to CPCs, we have done safety systems for monitoring, accident monitoring, safety parameter display and others. The thing I'd like to point out is that for the CPC situation, we have had basically more than 800 software modifications, what we call software change requests, since the installation at ANO-2.

Ninety-nine percent of these software change requests have been functional design changes, not software errors, not software bugs. These 99 percent are the types

of things that would show up in a hardware-based system as well as a software-based system because the root cause is the functional design process, not the implementation process.

We do a very good job of writing software to wrong requirements. Software runs the way the wrong requirement told it to run. The other point that I'd like to make is that in all of the operational experience that we have with the CPCs and where it did things that we didn't intend it to do functionally, none of those have resulted in failure to trip conditions. All of the software errors have been what we would call spurious trip conditions.

The point that I'd like to make on this slide is --

MR. KERR: Is that because you were clever in your design or was it just a fortuit to the circumstance?

MR. SCAROLA: No. I think it's inherent in the fail safe nature of the design. What we do is we force the system to go into a trip condition in any situation that you might call a system not knowing what it should do. So we force it to trip under any failure situation.

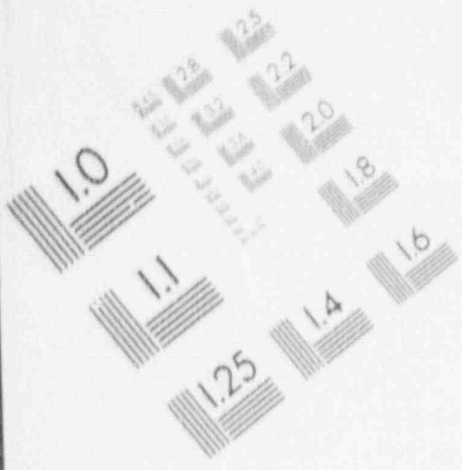
We have never had a situation where it would not trip.

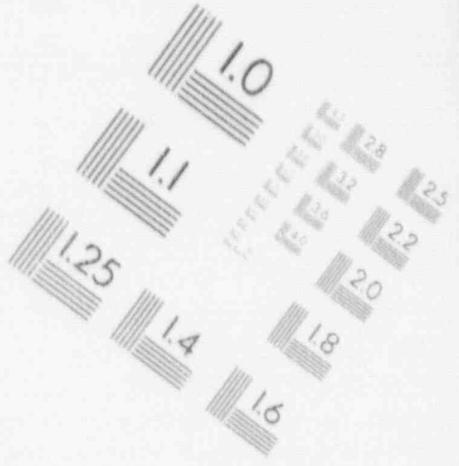
MR. KERR: Thank you.

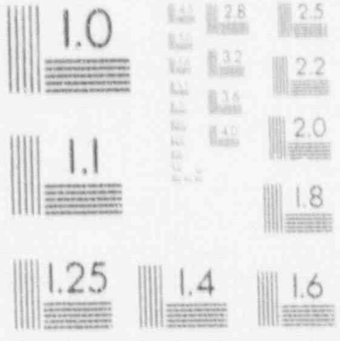
MR. LEWIS: When you say failure, you mean

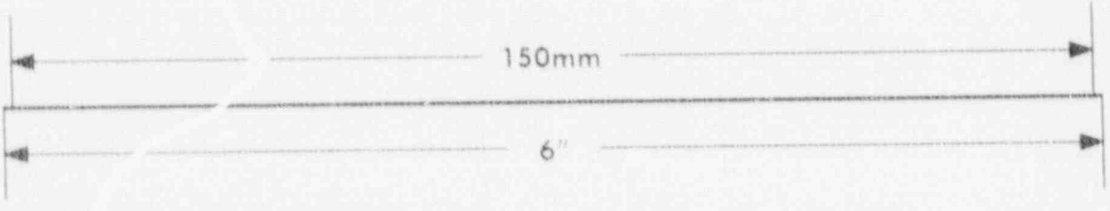
1

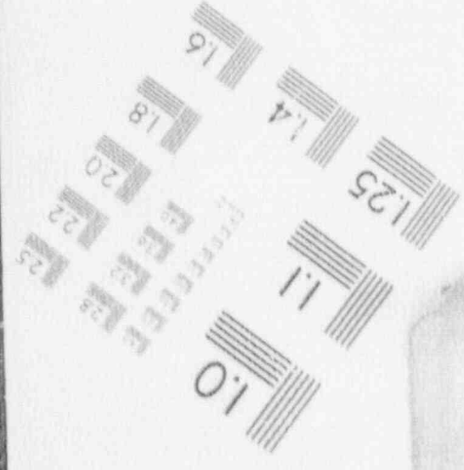
IMAGE EVALUATION TEST TARGET (MT-3)

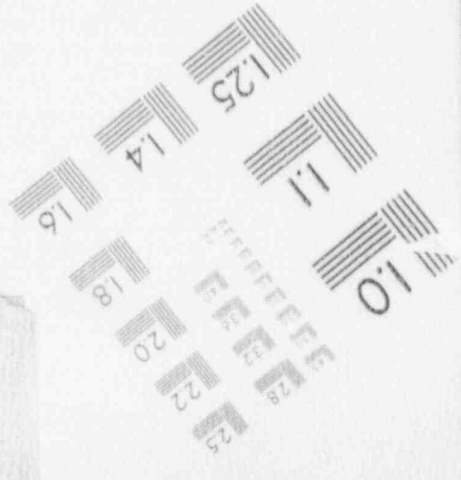












software failure, is that right? When you said in any failure situation.

MR. SCAROLA: Yes. I mean software failures in this slide. I would have to go back and research whether or not I could say that same thing about hardware failures, but I believe it's the same with hardware failures.

MR. LEWIS: Because the question of the Rancho Seco event came up a little earlier and this was an example of a place in which a failure in a power supply resulted in not a software failure, but in incorrect inputs to the software which then did what it was supposed to do, and nearly brought on a really monumental accident.

MR. SCAROLA: I think that you're really reemphasizing my point. We're looking at software as a potential introduction of new failure modes into a system when, in fact, the hardware relationship to software is probably more dominant and the functional relationship is more dominant.

MR. LEWIS: We don't disagree about that. Paul?

MR. SHEWMON: With regard to the 99 percent functional design changes, not software, does that, again, reemphasize your point that it's the interaction of the hardware --

MR. SCAROLA: No. It's the interaction of the functional designer to the system designer. In other words,

what I'm saying here is that 99 percent of these 826 changes were functional design algorithm changes where we decided that the algorithm was way too conservative and we had to relax the requirements.

We were getting spurious trips in situations when we should not be getting trips. So the root cause of these changes are functional design changes, not software/hardware coupling at all.

MR. SHEWMON: I guess that's too subtle for me to see it.

MR. WILKINS: He's saying it's a setpoint.

MR. SCAROLA: Yes. Maybe spurious is the wrong word. What I'm saying is the algorithm executes, makes a decision that says you should scram. A functional designer went back, looked at that algorithm, did some analysis and said if we're really in that situation, we don't need to scram, so let's change the algorithm.

MR. CARROLL: Unnecessary scram.

MR. SCAROLA: Unnecessary would be a better word. Spurious is not the correct word, I'm sorry. That is the background. Let me say that in addition to safety system software, we have been designing software-based control systems. Our first installation was at SONGS. We have installations at LPNL and all subsequent plants beyond that.

Some of them are extremely difficult installations

where we used software-based systems right next to the power supplies that run our mag jacks. So when we talk about harsh EMI environments and the effect of EMI on hardware and software-based systems, we have extensive experience there, as well.

Now I'd like to talk for a minute about the software design process and the software documentation process that we use. First of all, as emphasized on the slide before, we need an early focus on establishing what would be really correct requirements and specifications. I think this is a problem that's recognized by the industry as the biggest contributor to the bad name that software has gotten in the industry.

It's not that the people who write software do it wrong, it's that the people who establish the requirements don't do it correctly. So we put a lot of emphasis on the requirements for the system both from a hardware and software point of view and what we call functional decomposition point of view. We decompose the functions down into small units so they're very understandable and manageable on a module basis.

We use standard coding and documentation techniques, things like deterministic coding. We have software standards, guides that tell the programmers what they can do when they program and what they are not allowed

to do, like going off in a branch and not returning.

MR. KERR: I guess I don't understand the relationship between the first bullet and those two things that follow it. The first bullet seems to say that correct requirements and specs weren't -- and then the second one --

MR. SCAROLA: What I'm trying to say is we establish functional requirements for a system, and I'll show it better on the next slide. Why don't I get through this slide, and then I'll show it better on the next one. Then we have a verification and validation program that I'll spend more time on, and then, lastly, extensive configuration control over the life of the product.

For example, I showed you on the CPCs, we track every CPC modification on every plant. That applies to both the purchased software -- when we buy an executive system, we know the rev of that executive system and we follow any modifications that the original designer of that software makes.

MR. KERR: That doesn't apply to your CPC software, I presume.

MR. SCAROLA: It does apply for CPC software. I'm sorry. Maybe you should ask your question again.

MR. KERR: I got the impression that you had developed your CPC software and it wasn't off-the-shelf purchased.

MR. SCAROLA: Yes. Excuse me. I answered the wrong question. CPC is not purchased software. CPC is CE custom software.

[Slide.]

MR. SCAROLA: Now maybe I can answer your question on the relationship of the requirements to hardware and software. Basically, the system development process, in a very simplistic format, looks like this, where we establish system requirements and these Vs are basically verification points.

MR. KERR: From this distance, to me, that looks very fuzzy.

MR. SCAROLA: Have you got this in your handout?

MR. CARROLL: We do.

MR. KERR: I'm looking for it.

MR. SCAROLA: That this says is system requirements and the V is a verification activity. My point is verification does not start at the software cycle of the system design. Verification starts at the requirements cycle. At this branch here is where we take the requirements of the system, we define a system description, but then we break those requirements into the allocation between hardware and software.

We basically say for this system, this is what the hardware has to do and this is what the software has to do.

Then we verify those with regard to the original requirements. In other words, are we meeting the original requirements that have been established by the designer.

The hardware and software then will eventually come together in what we call system integration performance testing. We do this test -- this is actually called validation. But then we verify the test results again. So that was the point I was trying to make in the last slide where I said hardware/software.

We establish requirements for the system, break them into the allocation between hardware and software.

MR. LEWIS: You used the words validation and verification very quickly there. You said you validate it, then you verify the results. Can you expand on what's meant by those words?

MR. SCAROLA: Sure. Verification is used to mean essentially the review process that the documentation reflects the requirements in the previous step.

MR. LEWIS: So the verification is about the documentation.

MR. SCAROLA: It's also used in the software coding process. You'll see over here that we write software code. We design hardware in parallel. But the software goes through module testing. That testing is a form of verification on a software module basis and we actually

review the test results with formal verification documents.

MR. LEWIS: How do you distinguish between validation and verification?

MR. SCAROLA: Verification is part of the step-by-step process. Validation is the integrated test at the end of the process where you say the integrated system goes back to the original requirements and meets what was established as an original requirement. Now, that's a validation test.

What I said is when you do a test, you write a test report. That test report is then verified again.

MR. LEWIS: I'm just trying to make sure in my own mind. So you use the words verification and validation to mean different things from what the computer science community uses them for. There's no problem with that. I just needed to know what was going on here.

MR. SCAROLA: I didn't realize that I used them differently.

MR. LEWIS: You do. They teach courses called verification and validation in which the words mean entirely different things.

MR. SCAROLA: I'll leave it at that.

[Slide.]

MR. SCAROLA: Something that might help to understand the V&V process is this table where, on the lefthand side of the table, we identify essentially the

documentation that is produced for a particular system. These are three levels of system importance in our design, from non-safety systems over to what we call safety systems

This basically identifies who produces that document and who reviews that document. We have an established program and that's applied to all systems in the design. We often talk about the level of independence of the verifier from the design process. This particular table identifies what the minimum level of independence needs to be.

Some of the things you'll see, we review the design process by the requirements team. Those people that set the requirements get involved in the review of the system descriptions and specifications, for example. Those people that establish the requirements actually do the final testing of the system.

This is a mapping that gives our system designers guidance as to how to do verification and validation for any particular system.

MR. LEWIS: I hate to interrupt you, because I am, in the end, going to tell you we're running out of time. It will be my fault. The people who do the verification are not the people who have written the software. You are, for the most part, buying commercial software and adapting it. I'm trying to understand how you're putting this together.

MR. SCAROLA: We do both. We write custom software and we buy commercial software.

MR. LEWIS: The people who review the system presumably don't review the commercial software because that's written and often hidden from you, I would imagine.

MR. SCAROLA: What they review is the configuration control on that commercial software. In other words, we impose configuration control requirements on the supplier. We go back and review the traceability and the history of that software.

MR. LEWIS: That's fine. In terms of the software you write yourself, the reviewers are not the people who wrote the software.

MR. SCAROLA: That is correct.

MR. LEWIS: And you know perfectly well that there is nothing harder in life than reviewing a code that someone else has written, even with the best of software practices. There's been lots of experiments and lots of mistakes get through the second and third iteration of third parties reviewing the software.

In the term configuration controls, do you include protection against sabotage?

MR. SCAROLA: I will address sabotage. Let me move on.

MR. KERR: Our previous speaker, Mr. Ichiyen,

seemed to indicate that they had difficulty arriving at appropriate standards of performance for software. That was the impression I got. Do you have standards for performance that are adequate or appropriate, in your view, and, if so, where did you get them?

MR. SCAROLA: The standards for performance are defined by the verification and validation team. They establish them on a system-by-system basis. We are not using an IEEE industry standard for software performance, for example, because we don't know when it exists.

From an industry point of view, we have the same problem. The way we handle that is internal system-by-system basis, we establish standards for performance.

MR. KERR: Thank you.

MR. CARROLL: How would I go about getting confidence that the standards that you've established internally are really the right standards or how would the staff get confidence of that?

MR. SCAROLA: The staff will review our verification and validation program. They will be able to audit the results of all the verification and validation steps. I think it's a process-related level of comfort. It's not a bottom line level of comfort that you can get.

I don't think you can establish software reliability at the bottom of the process. I think you have

to establish it throughout the process.

[Slide.]

MR. SCAROLA: Another contributor to software reliability is segmentation. What we mean by segmentation is that we take a system and we break the system functions into smaller units to basically execute on smaller processors. This allows complex software to become simpler by breaking it into little pieces to execute on small machines. This adds a level of defense-in-depth, not the final level of defense-in-depth, against common mode failures because it introduces functional differences between the different processors that have to really execute the software.

It introduces software coding differences. Because the machines are running asynchronously, the execution times are different. So they're exposed to different real world conditions. Because no one machine is the same as any other machine, it does introduce hardware differences, as well. So by segmenting functions, you do get some level of protection against common mode failure.

Another thing that segmentation does is it partitions the more probable failures, which are actually hardware failures, into manageable units.

MR. WILKINS: Of course, there's a price to pay, isn't there? You have to cement the segments back together

again somehow.

MR. SCAROLA: To a certain extent, yes. I think I can explain in the case of the protection system what segmentation means fairly simply.

[Slide.]

MR. SCAROLA: This table summarizes, on the lefthand side of this table are all of the events, the design basis events that our protection system is designed to protect against. Then across the top we have all the signals that the protection system monitors; basically, all of the reactor trip conditions.

The numbers are the processors that actually handle that trip function. So if we look at one of these for a simple case like a feedwater temperature decrease, we're monitoring Steam Generator 1 pressure by Processor No. 1, the second steam generator pressure by Processor No. 2. In this particular case, we will also get a trip from the CPCs on DNBR.

So for this particular event, there are three machines running asynchronously, all trying to protect the plant. That's what we call segmentation. Now, I agree that these three functions have to be cemented back together and the protection system does do that. It basically says you should get a trip on any one of these.

What makes it a little more complicated is we do

look for like coincidence among redundant safety channels, such that the A channel can't say I have a trip on DNBR and the B channel say I have a trip on steam generator level, and then you end up with a plant trip.

So we do look for like coincidence and that does force inter-channel communication.

MR. CARROLL: What is VOPT?

MR. SCAROLA: VOPT is variable over power trip.

MR. KERR: When you finish this process, are you able to predict with some degree of confidence the probability of failure per demand to trip?

MR. SCAROLA: We do put a number on it. That is part of the availability analysis for this system and then that gets factored into the PRA.

MR. KERR: I don't understand the phrase "we do put a number on it."

MR. SCAROLA: Well, what you said is to what level of confidence --

MR. KERR: No. I said can you predict to a reasonable level of confidence. Can you give me a number, some indication of --

MR. SCAROLA: Yes. By the standards that we use, we can put -- we do put numbers on these things, and the reason that I qualify that is the industry right now has difficulty putting a reliability number on certain

unreliability contributors, such as human error and software reliability.

So to the extent that we can put numbers on meantime between failure, meantime to repair, and we can analyze failure modes and effects analysis, we do put numbers on these things. The other contributors are not well handled yet.

MR. CARROLL: But wouldn't that be the same number if this were an analog system?

MR. SCAROLA: No. Actually these are higher numbers because the MTBFs on these systems are much lower than analog systems and the meantime to repairs are much shorter.

MR. CARROLL: So it does include hardware other than sensors.

MR. SCAROLA: Yes. The sensors are, in fact, the same. There is no difference. I said segmentation gives a level of defense against common mode failure. I'd like to talk about another level of defense, and that's diversity.

[Slide.]

MR. SCAROLA: In operating plants today, we have significant diversity. It's not necessarily by design, but rather by the nature of the analog technology that we use in that if you have an analog circuit that has to do a particular function and then you have to define another

function, it usually takes a different analog circuit. So there are many different types of analog circuits.

It also relates to the number of people, the number of subcontractors that have gotten involved in the control complexes for nuclear power plants. This excess of diversity does give you a lot of defense-in-depth against common mode failure, but it may actually detract from plant safety because we have problems training maintenance personnel.

We have difficult repair times, long repair times because of that. We, of course, have spare parts availability problems. Spare parts availability is becoming even a bigger concern now with the obsolescence of analog --

MR. KERR: I want to applaud somebody who has the courage to question the gospel of diversity.

MR. SCAROLA: I'm sorry? I didn't hear the question.

MR. KERR: I want to applaud you for having the courage to question the gospel of diversity.

MR. SCAROLA: What we do in NUPLEX 80-Plus is we maximize standardization. We use standardization to the maximum extent possible, but we do maintain a minimum level of system diversity to offer what we call the final defense against common mode failures.

We employ diversity as a minimum in all software

based components of our systems. We do also employ it in some of the hardware based components of the system where required by rules such as the ATWS rule.

[Slide.]

MR. SCAROLA: To give you an idea of what that means in NUPLEX 80-Plus, on the lefthand side of this slide, I identify major functions. Over here, I identify Design Type 1 which is System No. 1 that accommodates that function and then Design Type 2 which is the diverse design system that can also accommodate that function.

What we do is we basically analyze that for every major function in the plant, such as reactor trip or for all of what we call critical functions, we have diverse means of accommodating that function or maintaining that critical function, whatever it might be.

We extend that as well to the information that the operator uses inside the control room. This may look somewhat complex, but, in its simplistic format, what this means is that Design Type 1 are all the safety systems in the power plant and Design Type 2 are all of the control systems in the power plant. So we force basically diversity between control and protection.

MR. KERR: Do you have different standards of reliability for the two?

MR. SCAROLA: Certainly. The protection systems

have higher standards of reliability because of the redundancy and single failure criteria that they have to meet.

MR. KERR: How much less reliable do you permit the control systems to be?

MR. SCAROLA: I can't answer that question. I don't know that we have a number that's an acceptance criteria. What I can tell you --

MR. KERR: To a certain extent, it seems to me that there is a good bit of artificiality in separating control and safety systems. The safety system is simply a control system that needs to be fairly reliable. Some of the other control systems maybe don't need to be as reliable. You haven't really thought much about the required liability of control systems.

MR. SCAROLA: It's not that. It's when you impose the requirements, irrespective of reliability, when you impose the requirements that we have to impose on protection systems to meet single failures and to have periodic testability, and when you extend that back to power supplies and HVAC and everything else which you don't do on the control systems, that's where you find that you have the major contributors to unreliability of the control systems.

MR. KERR: But, it seems to me, unreliable control systems can increase risk and can increase risk

significantly. If you look at LERs and other incidents, you find case after case in which you get trips because of an unreliable control system.

MR. SCAROLA: I won't disagree, but we are making the control systems orders of magnitude more reliable than we made them in the past. We do do reliability analysis on all of our control systems.

MR. KERR: You said you were making it orders of magnitude more reliable than something, and I didn't --

MR. SCAROLA: Than what we did in the past in control systems.

MR. KERR: Thank you.

MR. SCAROLA: I do have a slide -- I probably won't have the time. When I talked yesterday, I talked about fault tolerance and what we do to have fault tolerant control systems.

MR. MICHELSON: Excuse me, before you leave that slide. Your alarm and indication uses multiplexers, I guess, to get the information to the control room. Do you use different multiplexers for Design Type 1 than Design Type 2?

MR. SCAROLA: Yes. Anything that relies on software in these systems is different. Multiplexer certainly relies on software. The data communication relies on software.

MR. MICHELSON: So Design Type 2 has a dedicated set of multiplexers to get all of its information. Is that right?

MR. SCAROLA: Yes, to a certain extent. For example, if I look at a specific parameter, and let me take an example of pressurizer pressure. We monitor pressurizer pressure with both Class 1-E sensors and non-Class 1-E sensors. The non-Class 1-E sensors come into the system that we call the process component control system.

They are multiplexed into the electronics by that system. The safety-related sensors come into the protection system. So at that level, the multiplexers are totally independent. So when we get up to the monitoring systems, we combine all the information on both sides of this line. This system shows both safety and non-safety. This system shows both safety and non-safety.

MR. MICHELSON: So it came in with dedicated multiplexing, but it was then combined at the display level.

MR. SCAROLA: Combined, but into different diverse systems. In other words, both of these systems feed this one and both of these feed that one independently.

MR. MICHELSON: But that's only in the control room.

MR. SCAROLA: And at the remote shutdown panel. Control room and remote shutdown panel. I should have

brought a block diagram. That may have helped. I don't have much time, so let me talk quickly about sabotage protection, since that was asked.

[Slide.]

MR. SCAROLA: Sabotage protection is an important issue. We handle it in several ways. First of all, we maintain configuration control during the design, construction and the operation of the plant. That's very important.

Second, we physically separate into separate rooms the four channels of our protection system, and those are separate from the non-safety system. I think there's a slide in your package on that. That's this next slide.

[Slide.]

MR. SCAROLA: What this basically shows is that there are four separate secured rooms, separate from a security point of view, for the four channels of the protection system. Those four are separate from the non-safety equipment room.

We further have room access and equipment access security alarms. In other words, when you go into the room, the room is alarmed even though it is under configuration or it is under security control, as well. But then once you get inside the room, you have to get inside a cabinet. Those cabinets are locked and, when you open the doors, that

is alarmed, as well.

The final protection against sabotage is that in every system we do a continuous program memory checksum. That checksum is reported continuously to the data processing system, which is our central plant computer. So if, for some reason, that memory in the machine is altered either because of an electrical fault or because of a maintenance error or even sabotage, the plant computer system will identify that there is a memory checksum error. So that will be an indication that something has gone wrong in that system.

MR. LEWIS: In cases where the change in the memory is intended, an update or something like that, there's some kind of personnel control associated with changing the checksum record.

MR. SCAROLA: Yes, in both machines.

MR. LEWIS: It's really a checksum that you use, not a CRC?

MR. SCAROLA: I say checksum in the simplistic sense it varies system-by-system.

MR. LEWIS: I admire your speed-up. Are you almost there?

MR. SCAROLA: Let me just show you one more slide that I used yesterday because I think it's important in understanding software reliability also. That's automatic

testing.

[Slide.]

MR. SCAROLA: Historically, we talk about automatic testing in the sense of what does the machine do to test itself, then we forget about software. So we look at things like are we able to read and write from memory; does the CPU run; can we do communications; but none of that tells you that when the software needs to execute, that it will, in fact, execute properly.

In our protection system, we include continuous on-line automatic functional testing, meaning that in the protection system, we force the input to go into a trip state. That propagates through the system. It executes the software algorithm as if there was a trip, but we do it in one channel at a time.

We do it very quickly so that it does not propagate in the event that you get a second channel failure. We do it in a way such that a valid trip will always get through, that the automatic test will not get blocked. That's another layer that we put into the design to enhance software reliability.

With that, I thank you.

MR. LEWIS: We thank you very much. Everyone has done well in terms of staying on time. In that case, with the power vested in me by the system, I'll give us a 15-

minute break. Come back in 15 minutes promptly.

[Brief recess.]

MR. LEWIS: Can we begin? I apologize that we're running even a few minutes more late. My understanding is that your talk is not proprietary and can be open, is that correct?

MR. REMLEY: That's correct.

MR. LEWIS: The two after it, both before and after lunch, do have proprietary parts. I've been asked if the people who are going to give those could consider the possibility of treating their proprietary parts separately so that the audience out here can stay for as much as possible of their talks instead of having to leave for the whole thing because there are a few proprietary parts. I'll leave that to their judgment, but that's a plea I've been asked to make and I have just made it.

You are Gil Remley?

MR. REMLEY: Yes, I am Gil Remley.

MR. LEWIS: Very good. We are yours.

MR. REMLEY: My name is Gil Remley. I'm with Westinghouse Electric Corporation in the Process Control Division. Presently I'm responsible for the design of the integrated protection system which is Westinghouse's generic protection system design, and also the design of the primary protection system for the Sizewell B plant in England.

The package I've given you contains the overheads which I was going to use, and also there are two papers attached to the back of that package. Both of these papers were given at IEEE conferences in the United States. I thought they were particularly relevant to the topic.

One is one software diagnostics or software and hardware diagnostics in our systems. The other one is on a protocol for interface between multi-processors within a system. I'll be talking about that in a little more detail as I go.

MR. REMLEY: What I'm going to discuss is the software design primarily associated with the reactor protection and control equipment. However, a lot of what I say will be applicable especially to the rest of the plant I&C for control and data acquisition and, to some extent, also for the equipment that's used for information display.

However, just so I can contain the topic right now, I am going to concentrate on the protection and control software designs. But if we want to, we can also ask questions in the other areas and I'll try to clarify the differences, if you want.

[Slide.]

MR. REMLEY: The basic equipment in this design is based on distributed digital processing technology, particularly microprocessor technology. We believe that

there are significant benefits to be achieved by the use of this technology. This is a list of those benefits.

The design that I will discuss is characterized by the following points. It's a modular design. It uses digital technology. High performance elements are used where necessary. It is distributed processing in the sense that the processing is physically distributed, as well as functionally distributed.

It uses data highway and datalink communications. It is physically distributable. There is a hierarchical architecture for the communication and data transfer within the distributed digital processing system. Extensive use is made of fiber optic cabling. The design is characterized almost completely throughout by being fault tolerant. There is a clean separation between safety and non-safety.

We've implemented improved control and protection algorithms. Presentation of information the main control room is done in context with navigational aides. That wasn't the point I'm going to get into a lot of detail on. As I said, that's in the upper part of the design. But I can attempt to answer questions there if you're interested.

MR. KERR: Is this navigation of neutrons?

MR. REMLEY: No. It means navigation of information. It means navigation through presentation of information. It's a way of accessing data or information.

MR. KERR: Thank you.

[Slide.]

MR. REMLEY: The designs that we've been working on Westinghouse have undergone pretty extensive licensing review to date. This chart depicts that. There are some high points on the chart. I think one is the original concepts that we developed were associated with the original design called the integrated protection system, which was a hybrid design. It used both analog technology and microprocessor technology.

In the United States recently we've had several applications of this type of technology in plants in the U.S. The South Texas plant was mostly associated with the display of safety information. The Prairie Island plant was a microprocessor-based digital feedwater control system. The Sequoyah plants are a replacement of the process protection racks.

In addition to that, we've been developing this design in conjunction with many countries around the world. In France, we had a joint program for the development of IPS and the SPIN system, which is their microprocessor-based protection system in the French plants.

In England, as I mentioned, we're applying this technology as the primary protection system on the Sizewell B plant. We are using it on the APWR plant in Japan. It

was applied on the Italian reference plant in Italy.

[Slide.]

MR. REMLEY: I'd like to start off with talking about the software process that we use for development of the software and these systems. The first step in our design process is what we refer to as a requirements capture process. I agree with the previous speaker that mentioned that this is a very important step in the design process; to try to capture the requirements of your system.

However, what we do at this point with our particular requirements document is to try to consolidate and structure the various requirements that we get from numerous sources because you get requirements from many areas when you're trying to put together a design of a protection or a control system. You get functional requirements from the functional designers. You get industry standard requirements.

All this needs to be organized and sorted and defined with respect to an implementation that you would have in mind for the system. I do believe that there are significant industry standards available on which to base your requirements on. I have a list of them later. I think there are many standards groups that have been working over the past 15 years in this area that have done work to establish requirements for these types of systems.

I think you have to go and interpret that within the context of what you're going to try to do. That's one of the key steps that we do in the development of our system design requirements document.

The next step is to produce a document which then defines your particular implementation given that you've set out to achieve these requirements. At this step, one of the key things to do is to modularize the design or partition the design between hardware and software. This does occur in our system design specification document.

So it's the coordination document between the hardware and the software, because there is an intimate relation between the hardware and software in these designs. One of the things it does is it establishes the architecture for the system.

MR. KERR: Is there some sort of process of performing that division of responsibilities between hardware and software or is that left to the judgment of the designer? Do you have a prescription for doing that? Have you gotten that far along?

MR. REMLEY: Well, a cookbook prescription, no. But I guess it's really based upon, in a large sense, the traditional way people approach these problems. Maybe I'll try to explain that in a second. And also the availability of certain technology at some point in time.

Traditionally, certain functions are handled in certain ways because of boundary conditions in the plant. For example, the sensors generally produce analog signals. So that ends up being a given in the system design specification. You could revisit that and say you want the sensor to produce a digital signal.

But if you take that as a boundary condition, then that starts to partition the hardware and the software in the system. The second thing is that you have to work with available technology. You're limited by available technology. So certain things that were implemented in software five years ago are now implemented in hardware. For example, cyclic redundancy check algorithms for datalinks.

They are now embedded into the chip that handles the protocol for the datalink communications. Before, that was something that would be handled in software. You do, I believe, in the practice, want to push as much of that functionality to the hardware as you can, because I think there are significant performance benefits and operational base benefits you can get from that.

So I guess it really depends on your interface boundary conditions, that establishes an awful lot, and also what technology, what available technology you have to work with at any point in time.

MR. KERR: Thank you.

MR. REMLEY: So at this specification level, you do come up with an architecture for a system in partitioning between hardware and software.

[Slide.]

MR. REMLEY: This just happens to be a depiction of a particular implementation of our integrated protection cabinets. You see that we partitioned it into several microprocessor subsystems. Last year at this time I explained the rationale for having two reactor trip and two engineered safeguards the trip logic computer and a nuclear instrumentation system, and some support subsystems added the communications and the automatic tester systems.

These systems in our design are multi-processor systems. That is within each subsystem, you have several processors working together to perform the function of the subsystem. Typically we have one host computer and then several slave computers. The slave computers are mostly oriented toward handling input/output functions that have to be handled at very rapid speeds.

So what you do is you offload the host processor in the area where very rapid performance has to be achieved by using slave processors and then providing a way to exchange information between the two. This is a way to minimize the need for interrupts in multi-tasking, which we

do not want to put into the design.

So what we have done is we've distributed the processing and offloaded the higher performance needs into slave processors.

[Slide.]

MR. REMLEY: In the protection and control systems, we have three slave processors: one associated with analog inputs. We use this because we want to do digital filtering to improve the accuracy and the speed of the conversion from an analog signal to a digital signal. Also, we have slave controllers for datalink and data highway interfaces.

MR. LEWIS: Why is the digital filtering called intelligent AD?

MR. REMLEY: It was just a term that the software engineers used.

MR. LEWIS: Advertising.

MR. REMLEY: It's really just a --

MR. LEWIS: Gotcha. I understand. I won't press the point.

MR. REMLEY: One of the attributes of the system specification is to do this partitioning, as I mentioned, between the software and the hardware. Another one is to capture the functional requirements or the functional design in a way that we believe it can be fed back to the

functional designer in a way that he can understand what the implementation is going to be in the system so that it can be verified in a relatively straightforward way.

[Slide.]

MR. REMLEY: So as part of this document, we also have a section that's associated with defining the protection and control functions in logic diagrams. This is an example of a logic diagram. It, in fact, is sort of a hybrid data flow diagram that explains the interface between the software and the hardware.

It also then defines, in a graphical way, in a high level way, the protection or control function that is to be implemented. I think this goes a long way into trying to coordinate the interface between the software engineer who is going to program this in the system and the functional design engineer who is specifying the functions and making sure that the software engineer has a correct interpretation of what the safety function is supposed to be or the control function before he proceeds with the design.

MR. KERR: What is a low partial reactor trip?

MR. REMLEY: Excuse me? Okay. A partial reactor trip, in our terminology, is the system is basically a two-out-of-four design and each channel set can produce one-fourth of the input to the final trip gate. We refer to that as a partial trip on a particular function. Does that

answer your question?

MR. KERR: Except you haven't told me what a low partial trip is.

MR. REMLEY: Low means that it is coming off of a low bistable. Low I think goes with the description of the function, not with the -- it's not an adjective for partial trip. Partial trip is something --

MR. KERR: So it's a low flow associated, is that right?

MR. REMLEY: That's right.

MR. MICHELSON: What is the significance of the dotted line from the INE converter to the digital converter? Just above your total dashed line, those are all dotted lines. That does that mean?

MR. REMLEY: That is being performed in hardware, not in software.

MR. MICHELSON: Performed in hardware.

MR. REMLEY: If you look at the next two sheets, you will see a coding of all the symbols. I just attached them for information. That will explain all the symbols.

MR. MICHELSON: I don't see the dashed lines on there, but --

MR. REMLEY: Okay. I can't say that it is. I haven't studied it well enough to know that it is.

MR. MICHELSON: What's an analog line mean from an

INL converter?

MR. REMLEY: This signal here is now a voltage level signal.

MR. MICHELSON: I got it.

MR. REMLEY: The basic point I want to make reemphasizes that this is a relatively straightforward way of depicting how the function is going to perform within the equipment. So that we minimize confusion among the people doing the design and also within the verification process itself.

[Slide.]

MR. REMLEY: Another requirement that we built into the design when we move into the software itself -- once you've established the specification, then you're in a position to define the requirements that you need for both the hardware and the software. We do that independently in separate documents at that point called hardware design requirements and software design requirements.

These requirements at this point tend to be functional in nature. They tend to be requirements about the function of the software in the systems. But in addition to that, we have additional requirements that are mostly associated with the system high level requirements that are the standards in the industry that you need to meet for software for safety-critical applications.

What we've done is produced actually a document which defines the software design constraints that we use then to go ahead and program the system after we have produced this document. I'll give you some examples of some of these constraints.

MR. WILKINS: What's that 414 IPS?

MR. REMLEY: That was the design -- if I can go without putting the original slide back up -- maybe I can find it.

MR. WILKINS: That's all right.

MR. REMLEY: It'll only take me a second. A lot of the original design constraints were established in this program in the late 1970s. The system that we produced, this hybrid prototype was called the 414 Integrated Protection System.

[Slide.]

MR. REMLEY: In this document, we speak to a lot of areas in the area of software design, areas where we feel that you need to place special attention and impose constraints so that you will get a design which is highly reliable and verifiable. These are the areas where we have addressed these constraints. I can pick out a few examples. We've discussed interrupts already and the constraint associated with the use of interrupts.

We need constraints associated with multi-

processing because we intend to use multi-processing in our solution. There are other constraints associated with areas like data bounding, and you need to do this to help your verification process so they know the limits on which they need to work to do the verification itself. So within the system software it bounds the data that it uses.

The concept of application versus system software, we do not use commercially-available system software. We use system software that we've developed ourselves and verified ourselves, but we have a definite concept of how we want to distinguish between the system software and the application software and how those interfaces work. I will talk to that a little bit more.

This gets into also the topic of code versus data. Each one of these would be a long discussion. I'm just putting this up right now to point out that in the process, what we've done is we've established constraints in all these areas to improve the reliability and the verifiability of the software design.

MR. KERR: Do you use quantitative reliability criteria?

MR. REMLEY: The basis of the integrated protection system with respect to reactor trip is ten-to-the-minus-seven failures per demand. What we have to show, I think, is that the software is not going to degrade that.

With respect to quantifying the software itself, however, we have no program to do that explicitly. We do look at the operation of the software within the environment of the protection system in our analysis.

MR. KERR: Thank you.

MR. LEWIS: When you say ten-to-the-minus-seven per demand, demand is what?

MR. REMLEY: It's a demand on the system to trip.

MR. LEWIS: A trip demand. Thank you.

MR. CARROLL: That's the whole system?

MR. REMLEY: That's right. That's the whole system.

[Slide.]

MR. REMLEY: When you apply the constraints, you can come up with I guess a lot of solutions. However, the solution that we've come up with is a fairly straightforward single loop within all the processors. This is a depiction of what happens in that loop. As you can see, quite a bit of activity goes on at what we call restart time, and that is the application of power to the system or going and manually resetting the microcomputer subsystem.

Then this process here is a loop that basically repeats forever, assuming that nothing ever fails or you never reset it again. As you can see, it's a straightforward operation. At this point, we run in either

a mode that makes this loop go at a fixed frequency, and we call that synchronization and it doesn't mean synchronization of the multi-processors, it means just making the loop run at a fixed frequency, or we can choose to have it run at a non-fixed frequency.

An example of where we have the loop running at a fixed frequency is like in the reactor trip groups and the engineered safeguards groups because we need an exact time base to do the dynamic functions calculation, like lead-lag. Where we run it at a non-fixed loop frequency is in areas where we don't need to have this time base available to us for calculations. An example of that is where we do trip logic. We don't need a time base at that point.

[Slide.]

MR. REMLEY: As I said, the software is divided into basically two types; one that we refer to as application software, this is an application by a particular subsystem that I showed you on the first cabinet diagram; and, generalized types of software modules. These fall into two categories. Computer services; for example, analog input processing or a computer service type module; or protection and control algorithms are also done in a generalized way.

An example of that would be a lead-lag. This is a protection and control algorithm. The interface then

between the loop that I showed you in the previous diagram, which is the implementation of the application code, is then with subroutine calls to these two types of lower level software modules.

So the application level software, the code is very straightforward and, in fact, mimics to a large extent the logic diagram that I showed you previously. So it's a straightforward process, then, of verifying this particular code against this particular diagram, which is the application-specific function.

[Slide.]

MR. REMLEY: Associated with these particular modules, the services modules and also the control protection algorithms, is data and this data is of two types. One is calibration data and the other type is configuration data. I'll try to define the difference.

Configuration data is associated with the configuration of the equipment. An example of configuration data would be how many analog inputs in the subsystem, how many datalinks in the subsystem. Calibration data, on the other hand, is associated with adjusting the function of the system or tuning the function of the system. The gain on the lead-lag or the reset on the lead-lag would be an example of calibration data.

What we've done is we've partitioned the software

so that this data is in tables separate from the executable code. The reason we've done this is to improve the ongoing maintenance of the system, because we would like to verify these algorithms once very thoroughly and then use them in different applications to gain a broad base of experience.

So it allows us to use them in a broad sense and it also allows us to do a very thorough verification job, and then apply them by using different configuration tables without having to go back and recompile these modules. So the application ends up being the straightforward calls that I showed you that are associated with the logic diagram and then the generation of these tables which is associated with the calibration and the configuration of the system.

Then this application software then interfaces to the modules, which we do not have to recompile after verification.

[Slide.]

MR. REMLEY: I'd like to talk a little bit about the process for the software. As I mentioned, once we have developed the specification which partitions the hardware and software, we proceed through the step of defining the explicit requirements for the software then, writing a specification for that software and then the code that does the implementation.

We follow a process for that which involves peer

team review. This is a detailed logic structure of how that process is done. You'll see that there are certain points where design reviews are conducted. This is not verification. This is design review, peer design review after the requirements stage, after the preliminary specification and then, at the end, before the release to verification.

MR. LEWIS: Just out of curiosity, do you have a preferred language for code writing?

MR. REMLEY: We have selected PLM-86 as the high level language. The reason we selected it was because of the support that came with that language associated with the microprocessor that we were using, which was the 886 family of microprocessors. It is an acceptable language in that it's structured and it supports the constructs that we need to do the job and the way we defined the constraints.

MR. LEWIS: Does this mean in particular that you can't hire programmers off the street without then teaching them to write in that language?

MR. REMLEY: No. It's not a very difficult language to learn.

MR. LEWIS: I know that, but it's not the one that most programmers are brought up with.

MR. REMLEY: Actually, it looks a lot like Pascal.

MR. LEWIS: I know it does.

MR. REMLEY: There isn't a whole lot of difference. I don't think there's a whole lot of difficulty.

MR. LEWIS: Yes. But when you say one language looks a whole lot like another language, that's an introduction to mistakes in programming. Just curious.

MR. CATTON: That's Intel's language, isn't it?

MR. REMLEY: Yes.

MR. CATTON: It's Intel's own language, I think.

MR. REMLEY: That's right. That's correct.

MR. LEWIS: That's right. It's just that most kids who come out of school now knowing programming have different languages in their background.

MR. CATTON: This PLM-86 is not a very common language.

MR. LEWIS: That was the point I was trying to make.

MR. CATTON: And it's not as simple as you think. You probably have used it, so you think it's simple, but it's not.

MR. REMLEY: I'm not sure I said it was simple. If I did, maybe --

MR. CATTON: Even if you know Pascal, it's still tough.

MR. REMLEY: I said if you knew other languages,

it was simple to pick up. What I was trying to say is if you already know Pascal, then the transition to PLM-86 is not difficult. That's what I was trying to say. I wasn't characterizing PLM-86.

MR. KERR: He didn't mean the typical ACRS member could learn it, Ivan.

MR. CATTON: I know one of them who can.

MR. CARROLL: Or even the atypical one.

MR. CATTON: What operating system do you use, the RMX?

MR. REMLEY: We do not use an operating system.

MR. CATTON: You don't.

MR. REMLEY: No. That's what I was trying to say.

[Slide.]

MR. REMLEY: This structure is the operating system, if you will. It's the way the software works. It's a simple chain sequence. So in that sense, we don't use an operating system or this is the operating system, however you want to look at it. It isn't an operating system in the sense of RMX-86. It's not a multi-tasking, interrupt-driven system.

MR. CATTON: There's got to be something between PLM-86 and chips.

MR. REMLEY: No. What there is are these generalized computer services that are modules written in

PLM-86 that are called by other modules written in PLM-86. They're subroutines to the application. Now, sometimes we have to write in assembly language. Some of these are written in assembly language because of the performance requirements --

MR. LEWIS: Assembly for what machine?

MR. REMLEY: Excuse me?

MR. LEWIS: Assembly language is machine-specific.

MR. REMLEY: It's ASM-86. Assembly language for the 8086 family.

MR. LEWIS: I see.

MR. REMLEY: But we tried to stick to the high level language, unless there is reason why we can't use it and we have to justify it internally before we go to assembly language.

MR. LEWIS: But the problem of finding programmers who can write in assembly for a particular machine is worse in spades than finding people who can write in PLM-86.

MR. REMLEY: It is more difficult to write in assembly language than it is PLM-86, yes.

MR. CATTON: Lots of hackers can do it, but most of the hackers don't understand PLM-86.

MR. LEWIS: That's correct. It's just that the problem -- the reason for hammering this point is that the problem of getting independent verification becomes much

more difficult if you're talking about obscure languages.

MR. REMLEY: That sort of brings me to the next topic, which is software verification.

MR. LEWIS: And at the risk of really being mean to you, you're not running out of time, but you're getting close to it.

MR. CARROLL: We started him late.

MR. LEWIS: I know that. I'm giving him a warning, not a knife in the throat.

MR. REMLEY: Thank you.

[Slide.]

MR. REMLEY: I guess to make sure we understand what goes on in context with respect to the software, we're talking about a verification and validation process in our program that's associated with all these steps here; system steps, hardware steps, and software steps. They're really treated equally by our verification program.

We bring in different specialists in different areas, but we treat them, at least in a high level sense, the same way.

[Slide.]

MR. REMLEY: This diagram here then shows the -- this is hard to read, I understand -- but it shows the appropriate verification steps between the various design activities. As I said, this includes the software, the

hardware, as well as the system. So actually the software gets exercised in three different areas. It gets exercised in the software verification, it gets exercised in the hardware verification, and it gets exercised in the system verification.

MR. LEWIS: Again, to repeat a question that came up earlier. When you use the term verification, what do you mean by it?

MR. REMLEY: What I mean by verification is the process of assuring that the requirements of one step have been implemented by the following step. In other words, it's a review to see that -- or test, it doesn't make any difference. It can be a review or a test to see that the step going from one activity to the next activity has been implemented successfully. So it's the step from going from here to here, it's the step from going to here to here, or from here to here. It's the step from going to here to here, and here to here.

In this case, we do testing. These steps are basically done by analysis and review. These steps are done by analysis and tests, and I was going to talk about that in a little more detail in a minute. And then finally bringing the system together, you test to see that you've integrated it properly, but then the concept validation applies to the fact that you take the final product and basically in some

way compare it back to the original basis, which is the requirements, either by test or by analysis.

You can make this concept smaller than just the system. You can talk about validation in terms of the software versus its requirements, if you want to.

MR. LEWIS: I understand, but you're using the terms verification and validation almost interchangeably.

MR. REMLEY: No, I'm not. I don't think so.

MR. LEWIS: You're not. But in both cases, and you'll tell me the difference in a moment, but in both cases you're speaking of the performance of the system against the specs under normal conditions. You're not talking about --

MR. REMLEY: No.

MR. LEWIS: You're not.

MR. REMLEY: Because the system requirements talk about what the system should do under abnormal conditions. It's part of the requirements of the system.

MR. LEWIS: I see.

MR. REMLEY: It's more than just normal conditions.

MR. LEWIS: But then there's a real problem in describing what you mean by abnormal conditions, because for any reasonable size computer system that have 100 inputs, you're certainly not going to explore the complete range of possible incorrect inputs for all 100 channels. I'm

inventing the number 100. I don't know what it is for your system.

MR. REMLEY: In the big picture of the program, the answer is yes. But trying to do it from the outside of the system looking in, I agree that that's not the intent, but the intent is to build up to a point where you can justify the number of tests you've run on the integrated system.

You do what you're requesting, but it's done in the verification of the software modules. It's done in conjunction with the fact that we have bound the data that's in those modules. Remember I talked about data bounding. One of the reasons we do that is that the software itself bounds the data that it will use.

Therefore, we know the domain in which to run the test because it's inherent to the software module.

MR. LEWIS: For example, the last speaker spoke of testing the system with randomly generated inputs. Do you do something like that as part of the validation and verification program?

MR. REMLEY: We tend to use engineered test cases rather than randomly generated inputs.

MR. LEWIS: So that means it's easier to overlook an unforeseen mode. That's a prejudicial comment.

MR. REMLEY: Yes.

MR. LEWIS: Please. I'm holding you up and I want you to roll.

[Slide.]

MR. REMLEY: As I mentioned, this may be a little out of order. Some of the standards that we use as our reference in basis are listed here, and our requirements. This addresses both the design and the verification activities.

MR. SHEWMON: What is IEC?

MR. REMLEY: International Electrotechnical Commission. This document has received a lot of work in the international arena for software for safety systems.

MR. LEWIS: Please continue.

MR. REMLEY: My point is there are a lot of standards out there already.

MR. CARROLL: Are they any good?

MR. REMLEY: I think they are, yes. I think what is required is, like I said, you have to look at the standards and you have to write down what you're going to do based on these standards. I think that's an important step to say this is the standard, this is what I'm going to do. So it's clear to everybody who is reviewing the program what the intent is.

MR. WILKINS: To what extent and how rapidly do these standards become obsolete? The ANSI-ANS business in

1982 which is starting to be a long time ago. You don't have any dates on the others.

MR. REMLEY: I don't think they really -- I think the technology will become obsolete faster than the standards. I believe that it would require updating from time to time. People gain more experience with the technology and I think you can go back and improve most of these standards, but I wouldn't say that there's something that should just be tossed away. Also, they have a lot of good work in them, a lot of good thoughts.

The problem comes with standards on how literal the standard is meant. I think the writers sometimes mean something as a guideline and then it may be interpreted by other people as being literal. This is where you get into some difficulty. Then the standards writers try to improve their wording. That's why I think you need a document that explains your use of the standard.

Then it becomes clear how literally you have interpreted a particular requirement or have used it as a guideline and, if you've used it as a guideline, these are the conditions under which you will comply to the standard, but at least clarify the application of the standard.

MR. KERR: This would be an analog perhaps to the Nuclear Regulatory Commission's use of regulatory guides to explain regulations.

MR. REMLEY: Yes, I think so.

[Slide.]

MR. REMLEY: When we get to the step between the code and the requirements and the specification, we use an approach for the testing which is a bottom-up verification testing approach. It's an approach which is trying to stretch the design over the possible ranges of use of the modules. So that then they can be used from the higher level with assurance that they'll operate properly.

[Slide.]

MR. REMLEY: Graphically, the way we go about this is with a set of tools and approaches that are represented in this model. The model contains two dimensions. One is a manual and automatic dimension, and the other one is a static and dynamic.

Manual and automatic means how much automation there is associated with that particular activity on the part of the verifier. Static and dynamic means whether the code is, say, just sitting on a piece of paper there or is actually executing in a computer system. So what we've done is we have sort of a multi-dimensional attack in all these areas and we do all these activities with the code.

We have also developed some tools for aiding in that process and these tools are down under the automatic end. The tools are listed. I think this is effective for

the modules themselves. As I've mentioned, we do tend to want to build the system out of a lot of verified knowledge.

It tends to lose its effectiveness when you get into an integrated system because this doesn't deal with the interfaces yet. The issue in the integrated system tends to be the interfaces. For that, the system level testing is what addresses that.

[Slide.]

MR. REMLEY: The final point that I wanted to cover is the software security requirements. The first point to understand is that the embedded code in the system is all resident in prong. So that it maintains its integrity over loss of power and can be restarted without any intervention into the system.

Then associated with that there is the periodic testing, the periodic functional surveillance testing of a safety system which assures that the software hasn't changed or at least can perform its safety function. There are built-in diagnostics, which, in our design, include software keying. We actually have an ID associated with every subsystem that's built into a hardware key that's then read by the software, and then the software also has to have the matching part of the key embedded in its prong.

So what this assures is that you cannot locate any software subsystem in the wrong physical location in the

cabinets. The actual prongs have embedded checksums which are continually checked to see that the -- you can read them correctly and that the content hasn't changed.

As I mentioned, we make extensive use of read-only memories. There is limited physical access to the systems. We accommodate door locks and, as a matter of fact, on the Sizewell design they have a very elaborate door-locking system called the Fortress Interlock System, which only allows access to one what we refer to as channel setter train at a time. It's actually a key interlock that only allows you access to one type of key at a time.

We've designed it so that we only need limited physical access to the equipment in any event. We don't need to be tuning the system as much as we did with analog systems. We have the integrated surveillance testing. So we've limited the need for physical access and we don't have any need for software access in the sense of reprogramming.

We do allow a few data items to be changed in situ and those are limited to an exact data item that we've predetermined that will be changeable from the point of view of system calibration in situ. But a lot of the protection setpoints are not in that area. This tends to be -- like the calimetric scaling is one of these types of numbers.

That concludes what I had to say. Do you have any more questions?

MR. LEWIS: We've been asking questions as we went along So we thank you very much for staying almost on schedule. My agenda says the we're now going to hear from Tim O'Neil of GE, is that correct? Is this closed?

MR. SIMON: No. I have no problem with leaving it open.

MR. LEWIS: That would be great. Thank you. In that case, we are yours.

MR. SIMON: He is giving a presentation to Dr. Murley today back in San Jose. My name is Barry Simon. I'm the Lead System Engineer for Safety System Logic and Control, which is our digital protection system. This presentation is from the standpoint of our digital safety systems design. I fully agree with the panel and with the other speakers on the fact that the hardware cannot be separated from the software.

So I will first present the general layout of our system architecture. I will try to make this faster because yesterday I made everybody go hungry by running over. So I'll try to finish earlier.

This first slide just really says that computers will do good things for you, which I think we've established already.

MR. CARROLL: Why doesn't anybody ever present a slide that describes the bad things computers will do for

you?

MR. SIMON: We only talk about that privately.

MR. CARROLL: I see.

[Slide.]

MR. SIMON: If you're not careful, they will. One of the main benefits is that we're trying to reduce, as a practical matter, panel volume in the control room. So everything is being performed in microprocessors essentially. Distributed processing we use for distributing the intelligence to various points throughout the plant from the reactor building to the control room and in separate processors.

We're using multiplexing to cut down the quantity of cable in the control room and throughout the plant, and fiber optics to reduce EMI effects in general and to have high speed data processing and much smaller diameter cable, much lighter weight. Continuous self-tests and fault localization is one of our key benefits in reducing common cause failure and in increasing availability by reducing meantime to repair.

The use of micro-electronics allows standardization, which is a key item in inventory control. Also, to reduce maintenance time, surveillance time off-line, we have computerized test equipment. The added functionality refers to the improved man-machine interface

and improved algorithms that we can implement and improved displays for the operator, for instance.

In general, reduce the overall burden on the operator and allow the use of touch panels, electronic switches, and advanced controls to benefit the operator.

MR. KERR: Any significant difference between this and the earlier systems in terms of sensitivity to an in-plant fire?

MR. SIMON: To fire?

MR. KERR: Fire.

MR. SIMON: The distributed processing limits the effects of fire. The fact that this is a -- we use multiple redundancy, four divisions separated system to reduce the effects of fire. It's entirely a two-out-of-four system. Each division can be bypassed individually and an entire channel can be lost.

MR. KERR: That could be said for a pre-computer channel, couldn't it? I was just curious as to whether there's some weak point that this might have that non-digital systems don't have or some strong point.

MR. SIMON: Of course, weak point could be if erroneous signals were generated because of something like overheating, something due to fire. The use of fiber optics is a great benefit in reducing and having more complete isolation.

MR. KERR: Thank you.

MR. SHEWMON: Are the fiber optics designed to take any higher temperature than the copper wire cable?

MR. SIMON: The fiber optic cable will have equivalent insulation to copper cable, and it can also be -- you might also use armored cable. Are you referring to the heat effects on the glass fiber itself?

MR. SHEWMON: Yes. I would think those would be less than on the copper, though I don't know how they deteriorate with rising temperature. Certainly burning off the insulation doesn't inherently destroy the optical fibers' capabilities.

MR. SIMON: Of course, when you deform the coherent structure of the fiber, you lose the data or correct the data. But there's data transmission checking to take care of that. The main benefit is that you don't have the short grounding and hot short problem.

MR. CARROLL: You did say yesterday that ultimately you also have continued with the hard wiring of a manual scram.

MR. SIMON: Yes, definitely. That's the diverse backup to this whole system. The entire digital protection system can go down and you would still have your manual scram capability. Plus, through the remote shutdown system, you have remote shutdown cooling capability because all of

ECCS is there.

[Slide.]

MR. SIMON: If time permits, I'll also show you, in addition to ABWR, the SBWR which is not in your handout. I have additional material. In the SBWR, the simplified boiling water reactor, we have tried to implement simplification in the protection system design also, because most of the safety features are passive in that plant.

Safety system logic, our digital protection system integrates all the safety features, both scram and the ESF functions, ECCS and all the auxiliary support functions. It is in today's terms an embedded real time data acquisition and control system.

The fault tolerance is achieved on the highest level through the four-channel two-out-of-four voting. Then within each division, there's further redundancy, which I will show shortly. The control room logic is coupled to the reactor building multiplexers through the essential multiplexing system, which is independent within each or the protection divisions and which is also redundant.

All of our output switching is solid-state now also. We've had experience with the solid-state power switching since the Clinton solid-state design, which was not microprocessor controlled, but was all solid-state.

[Slide.]

MR. SIMON: Just as a quick overview, I presented this slide yesterday, also. This is the basic architecture of the system. The traditional sensor input are the usual analog sensors. Digitized at the remote multiplexer units, this is located in the reactor building in what we refer to as clean areas, temperature controlled and no radiation.

That data from the common network goes to the control room SSLC logic where there are three separate trains of logic. The top one, the top single train dedicated to the fail safe functions, reactor protection system and main steam isolation valve. The other two trains are engineered safety features.

We've divided those so that a loss of any one of the trains will not disable the entire decay heat removal mechanisms. These are also multiple in the four divisions. There are three divisions of ESF. Additional redundancy within each division or the dual processing at the output to prevent software or hardware failure from producing a trip, an initiation signal of ECCS, for instance.

This is two-out-of-four within each processor and there's two-out-of-two voting. So you have to have both channels in complete agreement in order to get an actual initiation signal at the output. We're dual through the entire multiplexing system. This section is out in the reactor building, also. So we have two trains just like

that.

ESF functions are multiplexed out, as was mentioned before. Fail safe functions, scram and main steam isolation valve closure are hard-wired. In addition to the manual scram which simply opens the power source for the valve solenoids, there is a manual divisional trip which is separate from the microprocessor control devices. So all this could go down or be bypassed and you would still have manual trip capability.

MR. MICHELSON: Could you, for clarification, just identify again the physical location of these blocks in the plant?

MR. SIMON: Right. There are four -- around the reactor -- in the reactor building, there are four established rooms. The emergency electrical equipment rooms, which have motor control centers and switch gear, which will also contain these remote multiplexing units. So the sensors are wired from their locations near the vessel to the rooms thereby where the analog-to-digital conversion occurs.

The fiber optics, then, these dashed lines, are then run to the control room in the control building, which, in the ABWR design, is an entirely separate building from the reactor building. The control room multiplexing unit then is physically in the control room; simply is the

receiving end of the data transmission path.

All of this equipment is in a single panel in each division. There are four of these four panels, one for each division, all in the control room.

MR. MICHELSON: They're all in the same room.

MR. SIMON: These are back row panels from the --

MR. MICHELSON: In the control room proper?

MR. SIMON: They're in another room from the main control console. They're separated -- they're in the same room.

MR. MICHELSON: But all of those panels are in the same physical location, no physical barriers between the panels.

MR. SIMON: There's physical -- the four panels are physically separated.

MR. MICHELSON: Four rooms?

MR. SIMON: But not in four rooms.

MR. MICHELSON: Physically separated means what, they're a few feet apart?

MR. SIMON: Several feet apart within the control room. On the output end, then, the control room multiplexer units are still in the control room, but in a second panel. There are fiber optic datalinks out back again to the reactor building. These remote multiplexer units are simply near these in the same place. Then the traditional contact

closure outputs and inputs are wired to the motor control centers.

MR. MICHELSON: The RMUs are adjacent to the RMUs or in those four rooms that you talked about where the multiplexers were located?

MR. SIMON: They're in the four rooms where the multiplexers are located.

MR. MICHELSON: Not where the RMUs are on the lefthand side of the drawing.

MR. SIMON: No. These are the same areas.

MR. MICHELSON: Same areas.

MR. SIMON: Same areas out in the reactor building.

MR. MICHELSON: Same room.

MR. SIMON: Right. In the same rooms. They could even be the same units, but for reliability we're separating them. So the sensors are input to separate units from the outputs. Because of the sensor reduction in the ABWR, which has two-thirds fewer sensors than a traditional plant, almost all the inputs to the system are actually contact closures from the motor control centers, valves, pumps, the interlock signals. You have limit switches, torque switches, position switches. That's by far the greatest input to the system.

Others are thermocouples and various devices for

each system. But the actual critical safety sensors are very few, probably less than dozen.

MR. MICHELSON: How are the RMUs protected against picking up upon faults, picking up higher voltages from, say, a motor control center?

MR. SIMON: The cabinets are shielded and grounded.

MR. MICHELSON: That won't stop 120 or 240 or 440 coming in on the wire, though, to the RMU.

MR. SIMON: We have surge protection for --

MR. MICHELSON: That's in the RMU?

MR. SIMON: In the RMU, yes. Of course, the fiber optics limits further propagation.

MR. MICHELSON: What kind of circuit protection do you put in to prevent a fault coming into the RMU? Say a 440 volt fault?

MR. SIMON: There is surge protection on the input line.

MR. MICHELSON: On each of those sensing --

MR. SIMON: Yes. The RMUs actually -- it's a combination. All of the RMUs are DC powered.

MR. MICHELSON: That takes care of it.

MR. SIMON: That's one thing.

MR. CATTON: Will they stop lightening?

MR. SIMON: Will they stop lightening? Once.

MR. MICHELSON: They will not stop surges, either, by the way.

MR. CATTON: I had the usual surge protection that you buy at the local computer store and lightening burned out the back end of my computer anyway. Went right over the top of it.

MR. LEWIS: Lightening is hard to stop.

MR. SIMON: Yes. We had experience with lightening at the Grand Gulf Station in Mississippi in the summer, where they have a lot of lightening. That plant gets a lot of direct hits which has damaged electronics. But it's always been pretty localized and hasn't propagated very far. But it has to do with the plant grounding system, too. The plant happens to have a very poor ground. You have to have the proper overall lightening protection.

[Slide.]

MR. SIMON: On reliability, I will just do this very shortly in order to get on to the software reliability. This is the hardware reliability I talked about yesterday. We have the defense-in-depth with protection systems separated from control systems because for the BWR, they're actually is no interface between the two.

We only send a reactor pump trip, recirc pump trip signal from the safety system to a non-safety system. After a scram, there is a control rod run-in signal that's sent.

But there are no signals that go from the control system to the protection system. And the redundancy, self-diagnostics.

The continuous operation refers to each division runs entirely independently and asynchronously.

MR. KERR: What sort of liability standards do you use for your control system?

MR. SIMON: The reliability is essentially equal.

MR. KERR: Equal to what?

MR. SIMON: To the protection system. It's really availability that we're going for more in the redundant safety system designs.

MR. KERR: Reliability doesn't include availability?

MR. SIMON: Well, certainly, but not always. For instance, the redundancy that I showed you with the two-out-of-two decreases availability, but it's reliable because the difference in the channels will actually --

MR. KERR: It's reliable in preventing false scrams, but it isn't as reliable in producing scram. I'm sorry, it isn't. If you have to have two-out-of-two instead of -- or two-out-of-four instead of one-out-of-four, you could miss some scrams, in principal.

MR. SIMON: In principal, you could. That's true. I forgot to say we have included a bypass for that, also.

So that one channel can be temporarily bypassed so that the remaining good channel could give you the correct trip while the other one is being repaired.

MR. LEWIS: What does it mean to say worst case design including environmental effects? Does that mean a combined magnitude earthquake and a 300-mile-an-hour tornado and lightening at the same time?

MR. SIMON: Those global effects are considered in the common mode failure design. But the worst case design refers to the power supply range, worst case range you expect for over-voltage and under-voltage and current. It does refer to the range of environmental conditions that you would expect, including accident conditions.

MR. LEWIS: Expected range or the worst case range?

MR. SIMON: Worst case range, loss of HVAC.

MR. LEWIS: I won't ask you to define worst case.

MR. SIMON: All our protection system equipment is mild environment. They're all in controlled environments with safety-related HVAC. The rest is use high reliability parts and, of course, qualification and then the integration during the V&V that I'll talk about.

Part of the basis of software reliability for the BWR is the simple repetitive or deterministic, as the previous presentation said, nature of the functions.

Essentially there are actually no calculations performed, but it's simply continuous reading of your sensor levels, determining whether they pass the trip thresholds. It is then just determined whether any two out of the four channels have passed the trip threshold.

You determine which of the sensors that involves and that causes your trip. There are no actual calculations performed.

MR. LEWIS: Is that regarded as a safety asset to not do mathematics? I'm trying to understand.

MR. SIMON: I'm just saying it makes verification and validation of the software simpler potentially.

MR. LEWIS: I'm not quite sure what is meant by not math intensive. That means you can add, but not multiply or what?

MR. SIMON: That means we do just add.

MR. LEWIS: You just add.

MR. SIMON: We don't perform complex algorithms. That's what it really means.

MR. CARROLL: I think the distinction, Hal, is that PWRs in terms of some of their trip functions, like DNBR and that sort of thing, have to do some calculations to see where they are, whereas a boiler, whether it uses digital or analog systems typically are you just reach a trip point and that's it.

MR. LEWIS: I appreciate that, Jay. I'm just trying to freeze in my mind the concept that doing mathematics is unsafe.

MR. SIMON: No, of course not.

MR. CARROLL: I don't think that's what he's saying. I think he's saying it's easier to check something that's on-off than it is to check something that involves a calculation.

MR. CATTON: I think it depends who does the mathematics.

MR. LEWIS: I'm hardpressed to really agree with that, but please go on.

MR. SIMON: I'm just saying that we think it's just simply easier. Of course, you can make the -- you can do all kinds of complex calculations and still make it safe and reliable through the V&V program. It's just that we think it's -- we're doing the V&V on much simpler code. That's all I'm saying.

MR. LEWIS: You're talking to somebody who had a delegation of students showing up in his office two weeks ago to say that a problem that had been assigned which required converting from meters to feet was impossible because the book gave a table which allowed them to convert from meters to yards, but they didn't know how to convert from yards to feet

MR. CATTON: It must have been Santa Barbara physics students.

MR. LEWIS: This was -- why don't you proceed here? Please proceed. Tummy grumblings will appear fairly soon, so you better speed it up.

[Slide.]

MR. SIMON: We also modularize the software within all those separate boxes I showed you. I should point out that the multiplexing is not intelligent. It's really a dumb multiplexer. It simply does the A-to-D conversion and transmits the message with a time-tag data to the control room.

Mainly for maintainability purposes, all the logic is performed in the control room where if something goes wrong, you simply replace a card within one of the boxes very quickly.

[Slide.]

MR. SIMON: Functional segmentation refers partly to the various trains within each division, where each of the systems, fail safe and not fail safe, and each of the different systems has its own software in separate modules. Full operating system not required means that we would use a non-formal operating system, as we mentioned previously, but just a real time -- essentially a real time for controlling the scheduling. Just putting the modules together and then

performing them, performing the functions. That's all backed up by the V&V program where I --

MR. MICHELSON: Before you leave the hardware, I wonder if you could clarify something you said during our Subcommittee meeting. If it's proprietary, fine, don't answer. You indicated the type of components you were using for the hardware; namely, the specifications that you were using. Would you care to reiterate that? I guess you know what I'm talking about. The temperature sensitivity of the equipment was discussed and you indicated the temperature rating of the components and the spec to which they were bought.

MR. SIMON: You mean the Mil Standard, the Mil Spec.

MR. MICHELSON: That's right. I didn't know whether that was considered proprietary that you were using Mil Specs or not. Could you tell the Committee the temperature -- I missed it.

MR. SIMON: As I said yesterday, we do specify Mil Standard 883(c) processing on all our components.

MR. MICHELSON: What was the temperature rating?

MR. SIMON: For the hermetically sealed parts, we're using 125 C.

MR. MICHELSON: 125 Centigrade. Okay.

MR. LEWIS: I noticed earlier you had Mil Spec.

Mil Spec also has minus 55 in it. Do you adhere to that?

MR. SIMON: We certainly hope so, except in Minnesota maybe.

MR. LEWIS: Is there a safety loss if you go to minus 50 instead of minus 55?

MR. SIMON: No. In fact, the components may run better when they're much colder. In fact, they're putting out small refrigeration units now for semiconductors to make them run faster.

MR. MICHELSON: I thought you also indicated that these components were adequate for inside of containment in hermetically sealed boxes without extra cooling needed. Is that correct?

MR. SIMON: Certainly.

MR. LEWIS: The reason for the Mil Spec minus 55 is for high flying aircraft. You're systems aren't going to be in high flying aircraft.

MR. SIMON: We hope not, unless the plant takes off.

MR. LEWIS: I am going to ask you to speed up, though.

[Slide.]

MR. SIMON: From the very top level through the detailed design, that V&V simply outputs at various points in the design to the V&V procedure. I will quickly present

-- I divided this into non-safety and safety just to show the major difference. This relates to non-safety verification, which is simply a series of design reviews. It's essentially the same pattern that has been presented before, separate from the top spec, the hardware/software spec, and the separate hardware and software development down to final integration.

For non-safety, we have a series of formal design reviews, but not the form sequential verification process.

[Slide.]

MR. SIMON: This is the same diagram as the previous one except for the added series of verification steps performed in each part of the design and at each software stage. We're separating V&V from actually the hardware steps which still go through the design reviews, but, for safety-related components, use the qualification process up until the integration stage.

Validation is the final step which was called the simulation test in the non-safety, but is the validation of the entire system to your system specs, your design and functional specs.

MR. CARROLL: What are the groundrules for somebody that does design review as opposed to somebody that does V&V activity? Can a person doing a design review work on the design?

MR. SIMON: No. The rules are somewhat the same, except the design review person would not necessarily have to be as knowledgeable of things like the software; ability to read the code, for instance. But the design review panels are simply independent review boards that are capable of understanding basically what the design is.

The verification people would have to go into much more detail to totally verify that the previous steps had been performed properly and that you were at a level where you could proceed to the next step.

MR. CARROLL: But both kinds of groups are independent of the people that actually did the work.

MR. SIMON: Yes. Through our internal guidelines. They are defined as being separate from the people who actually perform the implementation.

MR. CARROLL: In the QA world, we always talk about the pressures of production on the management person to whom a QA organization reports and so forth. Is there any distinction in your scheme of things in terms of who the design review people can report to versus who the verification people can report to?

MR. SIMON: No. There is some limited rules to that effect in that your reviewers are not supposed to be your management or your subordinates, but peers from other groups or management from other groups. That's the only

real limitation. Unless only those other people by justification are fully -- are the only ones that can do the work, that can do the review.

[Slide.]

MR. SIMON: The review process; the V&V program I've summarized is the one we submitted to the NRC for GE's present line of safety-related controllers, the NUMAC line of controllers. This would be extended to ABWR development. The informal reviews are not documented necessarily. They're just the day-by-day design process.

Independent design verification are those steps on the previous slide during the sequence of design. Now, the baseline reviews are formal reviews for establishing performance to the plan, essentially, the software management plan and to ensure compliance to performance specs to the high level code.

This review would ensure that the detailed code conformed to the high level code. These reviews are to review the actual test methods and review methods. They're always formally documented. The other part are just the formal testing steps, down through formal testing, release and methods for changing software, controlled changes of software by repeating various parts of the validation.

MR. SHEWMON: When you change the software in a plant, how is that done? Send a disk, send it over a

telephone line?

MR. SIMON: No.

MR. SHEWMON: Send PROMS?

MR. SIMON: No. We've also done the changes at the factory and then sent the PROMS to the site or installed them ourselves. But it is in all in PROM.

[Slide.]

MR. SIMON: Sabotage protection, I will just summarize very quickly. We have the usual physical security through the secured control room, the locked panels. In the reactor building, that's considered a vital area that also has card reader access to all the separate rooms.

MR. MICHELSON: How do you protect the PROMS before they ever get to the plant if all the changes are made back at the factory, if I understood what you said earlier correctly? Is that true that you make any software changes back at the factory and the PROM is --

MR. SIMON: That is correct.

MR. MICHELSON: -- sent to the plant?

MR. SIMON: That's correct.

MR. MICHELSON: So how do you protect them at the factory and during shipment and so forth?

MR. SIMON: Well, the software is --

MR. MICHELSON: Somebody can -- how much equipment would it take to program a PROM in a little different way as

opposed to doing it at the factory? Probably very little.

MR. SIMON: The safety software in PROM --

MR. LEWIS: They cannot be redone.

MR. SIMON: They cannot be redone.

MR. LEWIS: We programmed that --

MR. SIMON: Right. It would be difficult because of the type of checking that actually the other presenters have mentioned, the checksum, CRC, that are stored. Somebody who changed it would also have to know to change that or may not have access to being able to change the checksum. That would be discovered as soon as it was installed to a system.

MR. LEWIS: I'm suddenly confused because what Carl was asking, as I understand it, was what happens after they're originally programmed.

MR. SIMON: He's talking about physical protection, I believe.

MR. MICHELSON: At the factory and during shipment. But during shipment is a non-problem, I gather, unless you have sufficiently sophisticated equipment, but how about back at the factory.

MR. LEWIS: No. It's not a matter of sufficiently sophisticated equipment. There are two kinds of PROMS. There's the kind you can reprogram after they've been programmed and the kind that you can't. I thought he said

you physically cannot.

MR. SIMON: And these PROMS you could easily erase and reprogram. That would be bad, of course.

MR. MICHELSON: So shipment is a non-problem. Only back at the factory do you have a security problem.

MR. CATTON: You'd just replace it.

MR. MICHELSON: But you've got to make one.

MR. CATTON: But making PROMS is easy.

MR. MICHELSON: If it's easy --

MR. LEWIS: You can replace it.

MR. CATTON: You just swap it. That's quicker than trying to --

MR. MICHELSON: Then you worry about shipment, as well.

MR. CATTON: What kinds of checks do you go through once you get to the plant with these problems?

MR. SIMON: Right. Yes. They cannot be just immediately installed in the system, but have to be --

MR. CATTON: Do they fully check out the code that's on them and everything else?

MR. SIMON: Off-line. They have to be checked off-line in the actual piece of equipment, which all have built-in surveillance testing. You can run them.

MR. MICHELSON: But you must be able to change them because that's what you do at the factory, is you make

changes.

MR. CATTON: They make a new one, Carl.

MR. LEWIS: They make a new one, that's right.

MR. MICHELSON: But that's how they make changes.

MR. CATTON: They burn it into a new one.

MR. MICHELSON: Yes, but they've got -- if you want to make changes, you've got to make a new one.

MR. SIMON: A plant under its own security system could burn its own PROMS if it had administrative controls to do that.

MR. LEWIS: Yes.

MR. SIMON: But that would be fully under the responsibility of the plant, then.

MR. MICHELSON: But when the factory sends a new one that has some changes on it, which I think it can do, then how do you know whether the changes are acceptable or not at the plant?

MR. SIMON: Of course, they are tested against a revised specification.

MR. MICHELSON: So somebody has to send them some kind of a document that says it's changed and then you recheck, and that can be sent independently of the PROMS, hopefully not together.

MR. SIMON: Right.

MR. MICHELSON: The only place then would be the

factory where you'd really have to be secure.

MR. LEWIS: I doubt that anyone at the plant could do a complete check of a PROM. So you depend on it being done at the factory and that physical security of the PROM in transit.

MR. MICHELSON: That would be best.

MR. SIMON: I think the rest of it is all self-explanatory. I'm probably finished.

MR. LEWIS: In that case, I thank you for accelerating the process a little bit. Therefore, we will have our lunch break and come back at a quarter after one. The next item on our agenda is EPRI. Does EPRI have to be closed? EPRI will have to be closed. So the first object after lunch will be closed session.

[Whereupon, at 12:20 p.m., the Subcommittees were recessed for lunch, to reconvene this same day at 1:15 p.m.]

AFTERNOON SESSION

[2:00 p.m.]

MR. LEWIS: Let's get started. We are now on the record. Our next speaker is Leo Beltracchi, is it?

MR. BELTRACCHI: Yes, my name is Leo Beltracchi. I am a member of the Research Staff, Human Factors Branch. I will be discussing the research activities, both our current and future programs on use of digital computers in nuclear power plants.

[Slide.]

The starred items here generally cover programs that we have or information that we have with regard to safety applications, not all of them but most of them are. We also have some other applications that deal with the man-machine interface. I just want to point this out, the cognitive aspects of the man-machine interface and performance measures, and example of which was published in NUREG/CR-5348, man-machine interface issues in nuclear power plants.

That covered a workshop where we pooled many experts in the area. The object of the workshop was really to propose experiments and guidelines, and we currently are in the process of pursuing some of the proposed experiments in that area. I have a few of these reports with me if you are interested. The basic talk will address the starred

items, and let me address the first one on that now.

[Slide.]

We have had contact with several foreign countries. For example, we have talked with AECB in Canada on the Darlington experience.

MR. CARROLL: When you say we, will you tell me about the team and its composition with respect to nuclear engineers and computer scientists.

MR. BELTRACCHI: Okay.

MR. LEWIS: You learn fast, Jay.

MR. BELTRACCHI: Let me address that in the very first one in Darlington. We went up to see AECB, Curt Azmis specifically, about a year ago. The staff was supported by Oak Ridge, personnel from Oak Ridge National Laboratory who were experts in software engineering. We also had members of NRR come up on that visit as well.

MR. CARROLL: Was a multi-disciplinary group --

MR. BELTRACCHI: Yes, you could characterize it as a multi-disciplinary group. We discussed the Darlington experience with Curt Azmis and we also met with his consultant, David Parnas and discussed in detail the problems that they had in the review of the code, discussed why they had to go into reverse engineering. Let me quickly define that. They had a problem in trying to understand whether the code met the specification, and they couldn't

quite determine how to approach that. They eventually went to a method utilizing reverse engineering.

They used actually, function tables that were developed by Parnas to identify all the functions in the code, compare those functions to the functions that were called out in the specification and, of course, there were more functions in the code than there were in the specification. The excess functions were unintended functions, and the licensee had to address the safety impact of the unintended functions. That they actually did do, and those issues were resolved.

MR. LEWIS: So, the real honest to golly computer scientist in this operation was Parnas?

MR. BELTRACCHI: That is correct.

MR. CARROLL: No one on your team was a computer scientist?

MR. BELTRACCHI: I can't speak to the extent that they were a recent graduate from a computer -- say from a college accredited course, but we did have computer people from Oak Ridge National Laboratory support the audit.

MR. LEWIS: You described them before as software engineers.

MR. BELTRACCHI: Yes.

MR. LEWIS: That's different from computer scientists. I think we are just trying to find out who the

players are in this game.

MR. BELTRACCHI: The person from Oak Ridge was Mr. Ned Clapp, and he has been involved with software --

MR. LEWIS: That doesn't help me.

MR. BELTRACCHI: -- for a rather lengthy period of time.

MR. LEWIS: What is clear is that you, yourself, weren't all that concerned in detail about what their background in computer science was.

MR. BELTRACCHI: Yes.

MR. COFFMAN: Frank Coffman, Research Staff. This visit and the other visits that took place up there was to find out and learn from their experiance. There was no audit, there was no inspection, there was no review in the typical regulatory sense. The people who went up there ranged everywhere from Commissioner Rogers to Jim Snezik, Leo Beltracchi, Jay Persinski, myself, Joe Joyce, Jim Stewart has been up there, consultants

There was a wide range of background, none of which am I aware have a computer science degree from an accredited college. We are looking at the regulatory issues associated with the use of advanced systems and trying to let those issues drive what we need in the way of regulatory programs, and in terms of regulatory staff to support those programs.

So, the direct answer is no, I don't know of any computer science person. I'm not sure that's exactly what we need yet because we haven't clearly defined the regulatory issues or having completed our definition of regulatory issues from the research side of the house. Maybe the regulatory side would like to address it. I am just trying to give you a clear picture of where we are and what the context of the visit was.

MR. LEWIS: Thank you.

MR. KERR: What is a regulatory issue in this context?

MR. COFFMAN: One of the things that we did was to conduct a survey, and it identified something like eight regulatory issues. I think you probably only need one as an example.

MR. BELTRACCHI: It's the cost verification validation.

MR. COFFMAN: Right, at what point do you draw the line in expending resources on verification and validation when you end up making a judgment at the end anyway.

MR. LEWIS: That's bearing in mind that we mean different things, each of us, by verification and validation.

MR. SHEWMON: One of the regulatory issues on this that has been around for a long time is, how do you certify

whatever goes into the plant is not endangering the public health and safety here.

MR. KERR: I am just trying to find out what the staff means by it.

MR. COFFMAN: They range, but certainly one example is that when you introduce the new displays as one of the pictures that was shown this morning of the Darlington displays, then you have all the human factors aspects that go along with the ability to match up the operator's understanding of the physical processes that are going on with the way the information is portrayed on the display. That's one.

MR. CARROLL: That isn't a software issue. We are sort of talking software today.

MR. KERR: No, I am just trying to find out what he means by --

MR. COFFMAN: One of the other ones is how do you rely upon automatic monitoring, surveillance and calibration in the equipment.

MR. KERR: Jay, I want to find out what he means by a regulatory issue, and I think I am finding out.

MR. CARROLL: Okay.

MR. LEWIS: It's also true that when you speak of human interactions that's really not a software issue, and I worry a little bit about looking under the lamp post; that

is, see your own personal expertise in the subject. To use Paul's analogy, if you are interested in issues that involve the public health and safety you wouldn't call a computer scientist to come bang on the steel of a pressure vessel and say see, it sounds pretty good to me or anything like that. That's just as true of computer issues as anything else. You have to know what you are doing in order to know what you are looking for.

MR. KERR: I am satisfied with the information that you provided.

MR. BELTRACCHI: The software issues certainly are going to be such issues as common mode error to result in the loss of the safety function. You are looking for software engineering practices that would lead to high integrity software, and those characteristics that would support that. I don't think you can go out and say I have a yardstick and this passes and this fails, because the technology is not to that point yet.

MR. LEWIS: I have to differ with you on that. In many cases it isn't to that point and in some cases it is. There exists verification and validation tests which are pass/fail tests on computer systems.

MR. BELTRACCHI: That's right.

MR. LEWIS: So, sometimes it is true that you have a yardstick. Knowing where you do and where you don't is a

non-trivial --

MR. BELTRACCHI: That's true, and I will address some of those issues in this talk.

MR. LEWIS: Please, we are holding you up. I think that pretty much summarizes what I wanted to say about Darlington. We have also talked with AECB about their experience with Bruce, and that appeared to be a quality control issue since the problem was identified in the previous version of the code and hadn't been fixed. It did appear in an operational sense, and we are continuing our communications with AECB on this issue.

I would like to discuss some experiences in France and Germany. I was recently on a National Science Foundation survey, and this reflects some of the experiences that we gained from that survey. The N4 series of plants is the latest series of French designs. They just recently announced that they were cancelling part of their digital I&C for the control room. When you read about the details of it and also looked at the material, it appeared that the front end requirements of that design had not been thoroughly completed. It was certainly symptomatic of that problem, anyway.

We also found out recently that there also appeared to be some organizational issues that contributed to that cancellation.

MR. MICHELSON: Does that mean they went back to analog control?

MR. BELTRACCHI: They were going to revert to an earlier design --

MR. MICHELSON: Was it still digital?

MR. JOYCE: Excuse me. This is Joe Joyce with the Instrumentation branch. We have with us today John Gallagher, also a member of the Instrumentation and Control System Branch. He will be giving you a five to seven minute status on the N4 design, so let's not waste time on this.

MR. BELTRACCHI: I also want to point out that the French are in the process of designing a microprocessor-based safety system for the N4 series. They are using a case tool called OST that was developed at Saclay. It is also being used by their licensing people to evaluate the code.

In Germany, KWU is in the process of developing -- also in the process of developing a microprocessor-based safety system. It's a ten year program. They are also use a case tool that they developed in-house. It's called space specification and coding environment. It aids a designer in specifying the requirements, and it also has some characteristics of automatically generating code.

The licensing in TUV Norddeutschland is one of --

MR. LEWIS: What is meant by automatically

generated code?

MR. BELTRACCHI: Automatic code generator, in a sense that what they do is, they can end up with a specification that is in symbolic form and can read it optically and generate code.

MR. LEWIS: The specifications in written form, in symbolic form, are written in what language?

MR. BELTRACCHI: They developed their own very own direct language.

MR. LEWIS: Okay, so --

MR. BELTRACCHI: It ended up being graphical symbols. Of course, you have to understand the --

MR. LEWIS: This isn't really automatic code generation, it is code translation.

MR. BELTRACCHI: Okay, fine.

MR. CARROLL: Did this team, we or whoever, also look at the experience with digital applications in the United States? We heard each of the vendors to one degree or another, Combustion in particular, talk about a considerable experience base. Has this group looked at what they have been doing?

MR. BELTRACCHI: No. I am reporting on my experiences with a two week survey that was sponsored by the National Science Foundation.

MR. CARROLL: All of this was that?

MR. BELTRACCHI: No. This particular area here and here reflects that and overlaps my duties at the NRC, so I felt it appropriate.

MR. SHEWMON: The purpose of the NSF group was to do what?

MR. BELTRACCHI: To evaluate the research and current nuclear I&C instrumentation activities within Europe and compare it with those in the United States. We didn't do a specific survey within the United States. Experiences of members of that team were drawn upon to --

MR. SHEWMON: I am just some surprised -- this was part of the engineering division of the National Science Foundation? You don't have to answer it.

MR. BELTRACCHI: Okay.

MR. SHEWMON: It just doesn't sound too much like what they normally do.

MR. LEWIS: The Science Foundation normally doesn't have in-house expertise.

MR. SHEWMON: They normally don't worry about the engineering aspects of nuclear reactors.

MR. LEWIS: You bet you.

MR. BELTRACCHI: I guess two years back they did a survey of Japanese technology in the nuclear field, and this was sort of a follow up study.

MR. SHEWMON: Good. I have nothing against a

competent group.

MR. KERR: There is a report, I take it, in preparation?

MR. BELTRACCHI: The report is in the process of being generated. There should be a report by mid-summer or early fall.

There is a licensing authority in one of the German states, TUV Norddeutschland. They are using a case tool called SOSAT to evaluate the code that is being developed by KWU, and I will address that later in this talk.

[Slide.]

The next program that I would like to discuss is a program that we began two or three years ago. It was review criteria for human factors aspects of advanced I&C. The contractor was Oak Ridge National Laboratory. The objectives are stated here. It is basically to develop review criteria for evaluating safety implications of human factors associated with artificial intelligence, expert systems and advanced I&C.

The initial objective of this program did perform an industry survey to define issues. These were reported -- the results of the survey were reported in NUREG-5439, which I have a few copies of if you are interested.

MR. KERR: Who is in charge of the program at Oak

Ridge?

MR. BELTRACCHI: That was in the I&C area, and that was Dwayne Fry's instrumentation and controls division.

MR. KERR: He is not responsible --

MR. BELTRACCHI: No, it was Dr. Robert Urig, was involved.

MR. KERR: Is Urig responsible for the work?

MR. BELTRACCHI: Pardon me?

MR. KERR: Urig is taking responsibility for this program?

MR. BELTRACCHI: Yes. The survey portion of the program has been completed. He is no longer -- I don't think he is any longer working on this particular project.

MR. KERR: Who is then?

MR. BELTRACCHI: I think it's Mr. Carter and Bill Kinney.

MR. KERR: Thank you.

MR. BELTRACCHI: There were both human factors issues and instrumentation controls issues identified from the survey. I would like to just discuss one or two of the I&C issues that came from the survey.

[Slide.]

We found that there was a concern with respect to the resources requirements for verification and validation of advanced I&C. This concern reflected itself in the fact

that it was very costly and it was a question of a trade off, being able to deny anticipated improvements that were available from the use of digital technology, for example, your diagnostics that you can build into digital technology that are not available or readily available to an analog hard technology.

Another concern had to do with what are the configuration control requirements for digital systems backfitted in nuclear power plants. This manifested itself into the security of the software, being able to protect it against viruses, the maintenance of the software, as well as configuration control issues.

MR. CARROLL: It's really more than backfitted, it's also in an original design?

MR. BELTRACCHI: Yes. The acceptance criteria for advanced I&C was also an issue. Certainly, operators were concerned that poorly designed and ill-qualified equipment coming into plants would present them with problems. The need to establish acceptance criteria to avoid those kind of problems is certainly a factor that came from the survey.

The last bullet on here was a rather important one. In the course of our survey we found that one plant related a situation where their plant process computer was polling the protection system, and in the course of doing that there was a sneak circuit that actually tied up the

protection system. When they understood the problem, they way they solved it was to make the protection system just a broadcaster of data such as there would be no two-way communication, not even a handshake. That way, the information could be acquired by the plant process computer without impacting the protection system at all.

MR. KERR: Are there acceptance criteria for non-advanced I&C?

MR. BELTRACCHI: I guess if you would consider non-advanced I&C current technology, I would have to answer that yes.

MR. KERR: Where would one find such acceptance criteria?

MR. BELTRACCHI: In the form of regulatory guides and general design criteria.

MR. KERR: They also would cover, it would seem to me, advanced I&C as well because they cover what one is willing to accept in nuclear power plants.

MR. BELTRACCHI: To some extent they do and to some extent they don't. For example this one you could look at as a form of GDC 24 if you like, separation between protection and control. However, the intent of GDC 24 when it was written was really propagation of electrical faults. This is the software version of it, if you would like. That certainly was an issue when we addressed the core protection

calculator system in its review.

MR. KERR: What you are telling me is that the existing criteria are incomplete; they don't really cover it.

MR. BELTRACCHI: Or, the digital interpretation of that has to be made very clear.

MR. LEWIS: Could I pursue that one for just a moment?

MR. BELTRACCHI: Yes.

MR. LEWIS: This was a radio link or telephone link?

MR. BELTRACCHI: No, this was a digital link to the computer and actually had a hand shake in it. That is how it was described to us.

MR. LEWIS: It had a hand shake, that's what I was looking for. So, it is not true that it is a one way system.

MR. BELTRACCHI: That is correct.

MR. LEWIS: There is information going the other way?

MR. BELTRACCHI: That is correct.

MR. LEWIS: Therefore, the illusion that this is a one way system is perhaps mistaken.

MR. BELTRACCHI: This is the solution.

MR. LEWIS: Pardon?

MR. BELTRACCHI: This is the solution. I wanted to say that --

MR. LEWIS: The solution is truly one way?

MR. BELTRACCHI: That is correct.

MR. LEWIS: Of radio link?

MR. BELTRACCHI: Broadcasting out. There is no--

MR. LEWIS: By how, with a wire?

MR. BELTRACCHI: No, it just repeatedly presents the data to be read.

MR. LEWIS: I am asking whether it is transmitted by wire or wireless?

MR. BELTRACCHI: No, I believe it is transmitted by either an optical link or a wire.

MR. LEWIS: There is no hand shake?

MR. BELTRACCHI: That is correct.

MR. LEWIS: There is no signal going the other way? I don't understand how it works then.

MR. BELTRACCHI: If you were cycling through -- if you periodically put out every one second, then the listener would have to be looking for --

MR. LEWIS: It can be done. I agree that it can be done, but one has to look very carefully. One often finds that even when people tell you it is one way there are synchronization signals or other kinds of things that --

MR. BELTRACCHI: That may very well be true.

MR. LEWIS: But that compromises the illusion of isolation that you may get from this kind of wording.

MR. BELTRACCHI: That may very well be the case. What we are looking at here is really how to interpret this or what kind of requirements the NRC should be coming up with respect to say a digital interpretation, GDC 24.

MR. LEWIS: Please go on.

[Slide.]

MR. BELTRACCHI: There is another project that we have at Oak Ridge. It is entitled Computer Classification. This was review and evaluate the adequacy of resisting regulatory guidance for computer-based safety systems; and, where necessary, recommend development of new guidance. This was a two year program. It was initiated in February of 1989. We currently have a draft NUREG and it is under review, but it does need some additional work. We hope to publish that later on this spring or early this summer.

[Slide.]

The next project I would like to discuss is the expert system verification and validation guidelines. The contractor for this effort is Science Application International Corporation. The objective here was to develop and document guidelines for verifying and validating expert systems. This is a joint project funded by NRC and EPRI. It's a two year program that was initiated in

October, 1990. We are progressing on that effort at this time.

MR. KERR: Which branch of SAI is responsible for this?

MR. BELTRACCHI: They are over in Tysons Corner.

MR. CARROLL: Why would you have a project like this on expert systems and not one on control and protection software systems?

MR. BELTRACCHI: I am getting to the one on --

MR. CARROLL: You do have one on that.

MR. BELTRACCHI: Yes.

MR. LEWIS: I would have asked the converse question. I would say how can you do this on expert systems because, again, the term verify and validate would suggest a degree of formalism or formality that expert systems normally don't have.

MR. BELTRACCHI: There is one school of thought that the contractor has related to us that it would be easier to verify expert systems because of the formal methods and the tools that are available to do that.

MR. LEWIS: He must mean something different by the word verify again, because an expert system is just a collection of rules.

MR. BELTRACCHI: True.

MR. LEWIS: I don't see what you verify about a

collection of rules.

MR. BELTRACCHI: You are certainly concerned about whether or not the rules are conflicting, and you obviously have the concern of the degree of completeness and boundaries of that knowledge.

MR. LEWIS: I understand, but that's not the sort of thing that lends itself to verification in the computer science sense. I guess we are using that word for different purposes today.

[Slide.]

MR. BELTRACCHI: The NRC is a member of the Halden Project, Halden Reactor Project. One of the software tools that is developing at Halden is SOSAT. This is a set of tools for software safety assessment. It is being developed at Halden because TUV Norddeutschland had contracted with Halden to do this work.

The functions of this case tool are listed here. It will do metrics computations. For example, it will calculate the volume and length metric, check for illegal instructions and illegal accesses. It does static analysis of code and dynamic analysis. One of the future functions that they want to build into the system is symbolic execution. They plan to develop an analysis module to compare the program functions with specifications. You can read that as reverse engineering.

They are working on this effort, but they have not achieved that goal at this time.

MR. LEWIS: I am missing a point somewhere. You are not -- I'm on the wrong page, that's my problem.

MR. SHEWMON: He's giving the right viewgraph. Would you tell me what metric computation means there?

MR. BELTRACCHI: Yes. The Halstad metric, for example, is a measure of the complexity of the code. What they do is combine operators and operans by some log rhythmic formula, and if it exceeds a certain value it is an indication that it may be fairly complex or too long.

They have been able to correlate -- it correlates weakly, if I recall correctly, with a number of errors in a program. There was no strong correlation. At least that is what they found at Halden.

MR. SHEWMON: I wonder if it was calculating volumes in cubic meters --

MR. BELTRACCHI: No. There are also other metrics such as McCabe metric, and that has to do also with complexity. It can do time analyses for portions of the codes, provided you have loop criteria. It is now being used by TUV Norddeutschland, as I mentioned earlier. We have communicated with this regulatory agency, and we are looking into the possibility of using the SOSAT tool for our own NRC assessments of software.

[Slide.]

In another Halden project they had a goal of establishing increased software reliability for safety systems. This was really a program on software test and evaluation methods. They approached this by having one organization develop a safety system spec. That was the Safety and Reliability Directorate in the United Kingdom. This specification was independently coded by three teams: one in Norway, one in Finland and one in the UK.

They looked into many features. However, I am only going to focus on one aspect of the study; that is, the fault finding strategies and test data selection. Let me quickly describe the test data types that they had.

[Slide.]

There were six types. There were two sets of deterministic data. That is, systematic data, the type that you would manually produce test cases from the specification functions like a requirements matrix that would list all your functions, and you generate a test case for each of those functions. They use plant simulation data which would be like scenarios from training simulators.

They also had four sets of random data. One was uniform distribution with an equal probability inside the data range; that is, they had temperature from 500 to 600 degrees. They would have equal distribution of cases within

that range. They had a Gaussian distribution, mean and mid-range of that data, like 550. They had uniform distribution at the boundaries, both at the high and low. That's 500 and 600, and the example that I just --

MR. LEWIS: What does that mean? I don't understand what a Gaussian distribution boundaries is.

MR. BELTRACCHI: That means it was Gaussian around the high value and the low value of the input data range.

MR. LEWIS: I'm sorry, two Gaussian's, one at the high volume and one at the low volume --

MR. BELTRACCHI: Yes.

MR. LEWIS: -- and then uniform in between?

MR. BELTRACCHI: No.

MR. LEWIS: Just two Gaussian's.

MR. BELTRACCHI: Separate. The latter.

MR. LEWIS: Two Gaussian's --

MR. BELTRACCHI: These are separate cases.

MR. LEWIS: I see. That's for any of the variables, whatever they are?

MR. BELTRACCHI: Yes. And, the Gaussian distribution at the boundaries.

[Slide.]

In evaluating test data efficiency for fault detection, they took and seeded each program with 62 faults, and then they tested each program back to back against each

other with the input data types. They were able to find all of the seeded faults. However, multiple data types were required.

What they found was that the most efficient means of finding these faults were through the uniform distribution inside data range and the Gaussian distribution, the boundary. The least efficient were Gaussian distribution, mean and mid-range, and systematic data. That sort of points out that systematic data is the type of data that most people use to qualify their programs. It says that it is necessary but not sufficient in terms of a test strategy.

MR. KERR: Do you think this single investigation is enough to demonstrate this general conclusion that you are drawing?

MR. BELTRACCHI: No, but let me finish my point. I would like to also point out that in the core protection calculator they used a test strategy that was very similar to the combination of these two. Yes, you can question --

MR. CARROLL: They, in this case, being Combustion?

MR. BELTRACCHI: That is correct. You can question the issue of yes, these are empirical data and there is not a great deal of data here to support that. But it sort of does imply that if you want to really have some

technique to look for unintended functions, you better consider more than just systematic data. That's the only point that I wanted to make.

[Slide.]

We have another project that is addressed toward Class 1-E digital computer systems. The contractor is through an interagency agreement with Rome Air Development Center. The work will all be done by SOHAR, Incorporated. The objective here is to conduct an industry survey and develop technical bases for regulatory guidance on design, development and test and acceptance of Class 1-E computer systems. It's a one year program in response to specific user needs from NRR.

The product of this effort will be a draft regulatory guide on design and development of Class I-E computer systems. It will incorporate the survey results and research results that we know of to date. That is our goal.

MR. SHEWMON: Is this aimed at the reliability of hardware or software?

MR. BELTRACCHI: It will be principally addressed toward software and those hardware elements that impact software.

MR. KERR: This will be software that is capable of withstanding a safe shutdown earthquake?

MR. BELTRACCHI: I guess we will have to try that and see whether it works.

MR. CARROLL: Who is this contractor?

MR. BELTRACCHI: The contractor is SOHAR, Incorporated.

MR. CARROLL: I know, but tell me about them. I have never heard of them.

MR. BELTRACCHI: Let me take and ask Herb Heck from SOHAR to address that, and that will be a direct way of answering your question.

MR. HECK: SOHAR is a contraction of software and hardware reliability. We have been in business 13 years. We service, among other things, the FAA advanced automation system, service NASA and the Air Force. We also have a number of contracts with the Department of Energy in the nuclear field. In 1990 we were selected as the small business prime contractor of the year for the West Coast Region.

We have approximately 20 professional, very high level -- about one-quarter of the staff is Ph.D.

MR. CARROLL: Do any of them have their Ph.D. in computer science?

MR. HECK: All of them, except myself. Mine is in engineering, because when I got mine there wasn't much computer science.

MR. CARROLL: Do you feel possibly you don't have enough nuclear engineers to handle this job?

[Laughter.]

MR. LEWIS: Is this a group that has been dealing with RADC, or is it a spinoff of RADC personnel?

MR. HECK: No, sir. We have worked with RADC almost throughout our existence, but we are a for-profit organization. We have a task order agreement for continuing support to RADC.

MR. LEWIS: It says RADC/SOHAR on the viewgraph.

MR. HECK: It means that it goes to the task order agreement.

MR. SHEWMON: In the presentation by the gentleman from Westinghouse, there was a slide that listed various ANSI and IEEE standards and V&V guideline codes and standards. I didn't hear you mention that at all.

MR. BELTRACCHI: No, I didn't but I can. I can address that.

MR. SHEWMON: Let me finish the question, and then it might be more efficient.

MR. BELTRACCHI: I'm sorry, okay.

MR. SHEWMON: My background comes more in the pressure vessel end of things, where the NRC has a policy of trying to encourage industry standards and influencing how they get done. Is there a policy like that now formed in

NRC or will there be, or do they feel there is little value?

MR. BELTRACCHI: I believe NRR is going to be addressing their activities and the standards effort; is that right, Joe?

MR. JOYCE: We will have discussion on IEEE 7-4.3.2, with its endorsements of Reg Guide 1.152. We will talk about some of the work that is going on with revising the standard. We are fortunate to have a staff member with us that is part of that team. At that time, if you can hold off maybe, he can help with the question.

MR. SHEWMON: All right. Thank you.

MR. BELTRACCHI: That concludes my portion.

MR. LEWIS: Thank you very much for your speediness. I admire your ability to do that, in the face of the harassment you got from the members. Thank you. I believe that we can have a break, which will begin now.

[Brief recess.]

MR. LEWIS: Let us reconvene the meeting. We are now going to hear what it's really all about from the NRC staff.

MR. JOYCE: Is that why they put me last. I am Joe Joyce. Good afternoon. I am with the Instrumentation and Control Systems Branch. This afternoon I will be talking about some of our early designs, lessons learned from the early designs, present activities and criteria.

Then we will have Jim Stewart, also from the Instrumentation and Control System Branch. Jim will talk about future applications, advanced light water reactor, our passive designs, some retrofits that are going into the operating plants. After him we will have Ray Ets, who is with Software Associates. Ray is our consultant, and has been since 1987. Ray will talk about verification and validation, and our review methodology. Yes, he has his masters in computer science.

Then we will have John Gallagher, also with --

MR. KERR: He is not a member of the NRC staff though, is he?

MR. JOYCE: I'm sorry, I could not hear you.

MR. KERR: He is not a member of the NRC staff, is he?

MR. JOYCE: John Gallagher is, at the present time-

MR. KERR: No, Ed.

MR. JOYCE: -- at NRR in the Instrumentation and Control Systems Branch.

MR. KERR: No, Ets is the one that I am talking about, the one that has --

MR. JOYCE: Ray Ets is not a member of the staff. He is with Smartware Associates.

MR. KERR: He is the one that has a masters degree

in computer science, isn't he?

MR. JOYCE: That's correct.

MR. CARROLL: Does anyone in ICSB have such a degree?

MR. JOYCE: Computer science, I don't believe we do.

MR. STEWART: This is Jim Stewart. If it helps, I am currently in a master of science for computer science degree.

[Laughter.]

MR. STEWART: I would like to note that many of my professors in the masters program do not have their degrees in computer science for similar reasons to Herb Heck's, there weren't any available when they got their degrees.

MR. LEWIS: Some of us who teach for a living are delighted to meet people who are in a program on something and learning something.

MR. JOYCE: One thing I would like to add to that, because we have been asked that same question by the Commissioners. We have been trying to recruit -- we have had interviews and we have recruited people within the Instrumentation Branch. I, personally, have probably seen over 30 and I know the Branch has interviewed over 40.

We made three offers to computer science folks and they turned us down -- not enough money. We are now

continuing to interview and we have not stopped. We recognize that both the ACRS and the Commissioners would like to see ICSB staff augmented with a person that understands code one's and zero's. Even though we have been doing it for many moons and our consultants, we are still looking. It's not as if we aren't going to get one.

MR. CARROLL: How about statistics?

[Laughter.]

MR. JOYCE: That's a different branch.

MR. LEWIS: Ignore my trouble making friends and go on.

MR. JOYCE: I am going to go pretty fast because we have a tight schedule.

[Slide.]

Our first review was the core protection calculator system, which was designed by Combustion Engineering. You heard a little bit about it this morning. That was the first computer system that was doing some complex algorithms that used six minicomputers and had 12 inputs to the reactor protection system. Two of them were digitized, the DNBR calculations and kilowatts per foot. The rest were analog, and there was a warm feeling about that.

I am not going to talk about the design or the configuration. What I want to point out is that the

licensee and vendor, from the time they started this design into implementation probably spent approximately 100 man years on this task. The staff alone spent 18 man years in its review effort of this. There was a major redesign required during this process. At the end, the staff ended up developing 27 positions on the core protection calculator, many of which are still used today.

MR. KERR: That's one and one-half positions per man year.

MR. CARROLL: In what form are these positions? Where would I go to look for them?

MR. JOYCE: You can start going to the document called Arkansas ANO-2 safety evaluation report. There are two chapters. Chapter seven is instrumentation and control system. There is an appendix that talks specifically about the review methodology and the review efforts by the core protection calculator. In there you will see the 27 positions, 27 positions like being watchdog timer. Your system will have a watchdog timer. Any failures that happen within the system, watchdog timer times out and you go to a failed state and things like that.

MR. MICHELSON: Did they get into the standard review plan?

MR. JOYCE: Did the positions get into the standard review plan?

MR. MICHELSON: Yes.

MR. JOYCE: No, they did not.

MR. MICHELSON: When you say in the future that you are going to follow the standard review plan it doesn't mean that you go back to Arkansas, it means you go to the standard review plan to see what is required.

MR. JOYCE: That's true.

MR. MICHELSON: Whatever was at Arkansas and became a position somehow is also a requirement in the standard review plan then, or it is no longer needed.

MR. JOYCE: It kind of takes two paths. It takes the path called evolution, evolution and ICSB review effort. As we go off and we do the reviews as we did this review and other reviews, in the back of our mind and our hip pocket in the top right hand drawer we have the 27 positions. We know what the positions are, we documented them in previous SER's, we know what the concerns are, and we can continue to do the reviews incorporating positions and re-evaluating positions.

MR. MICHELSON: What puzzles me --

MR. JOYCE: Do they ever get to the standard review plan, no they have not.

MR. MICHELSON: How do they get into the improved evolutionary reactors? What positions do you have on those that might relate to solid state control?

MR. JOYCE: I am going to let Jim Stewart talk about those plants and those positions.

MR. MICHELSON: Okay, thank you.

MR. JOYCE: You are welcome.

MR. CARROLL: Tom, could you get the pages out of Arkansas that he is talking about?

MR. JOYCE: If you can't find them, call me and I will provide them to you or at least give you the documents.

MR. CARROLL: Probably most of the members would like to see what the 27 positions are.

MR. JOYCE: As a matter of fact when I get done here, I have the 27 positions xeroxed in case it came up. I will hand them to you at the end of the presentation.

MR. ROTELLA: Thank you.

MR. MICHELSON: It may be better to hand them during the presentation.

MR. WILKINS: No, because then he would talk about each one of them, and that would take 27 minutes.

MR. CARROLL: Or 18 man years.

[Slide.]

MR. JOYCE: The next review that we had was by Babcock and Wilcox. It was a reactor protection system II. This used four microprocessors and it had ten inputs to the reactor protection system, three of which were digitized. Once again, the DNBR kilowatts per foot, and there was a

flux offset I believe.

The results of this review, there are 14 errors found during integration testing which indicated a lack of detail review prior to testing.

MR. CARROLL: Found by B&W?

MR. JOYCE: Found by our audit team, B&W, their consultants and our consultants. We then had a contract with Boeing to also go off and look at some software modules. We went in and took samples of software modules, and Boeing did a sneak circuit analysis on it. As a result of that analysis there were nine software documented errors but there were no sneaks in the circuit.

MR. MICHELSON: What vintage B&W plants did this appear on?

MR. JOYCE: This is the Belefonte.

MR. MICHELSON: That's what I thougnt.

MR. JOYCE: Which leads to the next bullet. Our review was terminated by the cancellation of Belefonte. It is my understanding that Belefonte is in talking to the staff about resubmitting their FSAR.

[Slide.]

You have also heard these words used this morning, 414 integrated protection system. This was a major change for the Westinghouse design. This was a distributed digital microprocessor-based system that encompassed reactor

protection system, engineered safety features and control systems. Both the ACRS and the staff had a number of concerns about this design. We were concerned about the adverse interactions between all these systems. We have concerns about sharing of a common sensor or signal. We are also concerned -- one of the concerns was common mode failure in the redundant elements and the degradation of the defense-in-depth concept.

We ended up putting together a review group to assess the defense-in-depth and diversity of the integrated protection system. Half way through this review effort it became obvious to the staff that we were emersed in details, and we were not going to achieve the goal of the task force in finding the integrated protection system acceptable at the level of detail in which we were doing the review. We were down at the component level.

We decided that we can't do this, we will not be successful. We had to take a different approach, so we stepped backwards and decided to take what we call the simple approach and assess the system architecture only. At this time we had to develop what we call the block concept. The block concept also had guidelines associated with the block concept.

In conjunction with the block concept and the guidelines, these gave the staff tools to go in and assess

the integrated protection system with respect to the acceptability of the system architecture to the guidelines. We documented that in NUREG-0493. As a result of that we ended up giving Westinghouse a PDA for RESAR-414. I am sure you already know that.

MR. CARROLL: You have a xerox copy of the nine open items over there too?

MR. JOYCE: Yes. The nine open items that we had on RESAR-414, many of them have been closed. Many of them had to do with -- in the SER we talked about ongoing design and ongoing verification and validation. Many of these nine open items ended up being put to bed through verification and validation, which leads into the next slide.

MR. MICHELSON: On the Belefonte situation, you never got to the point of reaching positions, is that right?

MR. JOYCE: That's correct, we really didn't.

MR. CATTON: This morning we heard from GE that there were our IEEE standards and one ANSI standard and an ICE publication 880. Does NRC require --

MR. JOYCE: Yes. We don't require -- we will talk about them. If I can hold off on that, is that okay?

MR. CATTON: Fine.

MR. JOYCE: If I forget, remind me or Jim Stewart, one or the other.

[Slide.]

What were the lessons learned from these early designs? The major lesson that was learned -- I shouldn't say that. The NRC management made a decision that the staff or the Agency itself could not afford the resources that were required to go off and do this type of review. They didn't put the burden back on the staff to go off and find other tools, mechanisms, methodologies, something else other than what we were doing on the previous design for the other designs that were coming in that were using microprocessor-based systems.

At that time we looked around at the aerospace, we looked at the military, foreign countries, and we even had some people on working groups. It was decided that the verification and validation methodology seemed like a pretty good tool to go in and assess software quality. It looked like it was a pretty good tool that could be applied to the type of designs that we have done in the past, and probably also will apply to designs coming in the future.

MR. MICHELSON: This was just for software?

MR. JOYCE: Yes.

MR. MICHELSON: How about the hardware. I thought you said you were getting bogged down on components on the hardware, but maybe I misunderstood.

MR. JOYCE: You didn't misunderstand, that's what I said. What we did was, we used a traditional review

method on the integrated protection system that didn't work. The traditional method was --

MR. MICHELSON: You dislocated the flow diagram.

MR. JOYCE: Right, and started working our way down and said we can't get here. We are swamped under, we have to do something different. There are too many elements in there where we can go in and do our single failure to common mode failure --

MR. MICHELSON: Since that was kind of a unique or new approach to trying to do these reviews, was it documented in the standard review plan?

MR. JOYCE: What?

MR. MICHELSON: This new approach, this approach of going --

MR. JOYCE: No, sir. It's documented in NUREG-0493.

MR. MICHELSON: In the future if I say the standard review plan defines what needs to be done, I guess you are going to address it later -- that clearly wouldn't help too well.

MR. JOYCE: The standard review plan keeps coming up. Our goal is to update the standard review plan. I said this year -- my boss cringes which I say that -- it is needed. It is something that we have not necessarily done over how long --1984 is the last version we did. There is a

real need for that. We have the material, we have the experience, we have some of the knowledge that should go in for the reviewers and industry to look at, to see things like NUREG-0493 that talks about diversity --

MR. MICHELSON: I am sure you are aware that a lot of people think that the standard review plan is what the Agency uses to review designs by. I think what I am hearing today is that it wouldn't work too well in this particular instance of the hardware.

MR. JOYCE: You are absolutely right. The things that we are reviewing, these designs, are not in the standard review plan.

MR. CARROLL: Our Committee letter on SP-90 said that very clearly; that they ought to get on with getting that standard review plan up to date.

MR. JOYCE: It's just not there.

MR. MICHELSON: It's just further validation of what we strongly suspected for the Westinghouse case.

MR. JOYCE: We ended up endorsing this IEEE standard with Reg Guide 1.152. The lessons learned from the early designs -- now we have something -- we have tool and a methodology, and now we can put the burden back onto the utility and licensee and the designers of this equipment to use such a mechanism called verification and validation, so all the staff has to do is go in and do an audit at the end

of the audit.

With these tools including NUREG-0493, the lessons that were learned were from the early designs.

MR. LEWIS: When you say all the staff has to do is audit, that means the staff has to look at what the licensee does in terms of verification and validation and make sure he did it right.

MR. JOYCE: Yes.

MR. LEWIS: That requires that one be able to do it better than the licensee could have done it.

MR. JOYCE: Not necessarily.

MR. LEWIS: How do you audit something without knowing how to do it better?

MR. JOYCE: We will get into that when we talk about that with Ray Ets. We have a whole dissertation on review methodology.

MR. LEWIS: You have a dissertation on it?

MR. MICHELSON: This is coming --

MR. LEWIS: The first slide I put up --

MR. LEWIS: You are going to talk about it later, is that what you are saying?

MR. JOYCE: Yes.

MR. LEWIS: Is that what you were trying to communicate to me.

MR. JOYCE: My wife accuses me of that so many

times, poor communication skills. Sorry about that.

MR. MICHELSON: How much time do we have for all of this?

MR. LEWIS: You do disagree with the assertion that in order to audit something you should know more about it than the person who did it?

MR. JOYCE: No, I don't agree with that.

MR. LEWIS: Okay, fine. I thought you did.

MR. JOYCE: No.

MR. WILKINS: He said yes, he does agree that he disagrees.

MR. LEWIS: Okay, let's get the signs of the yes and no straight. You do disagree with my assertion that in order to audit something you ought to know more about it than the person who did it; you do disagree?

MR. JOYCE: Yes.

MR. LEWIS: There's nothing wrong with disagreeing. One of us will turn out to be right.

[Laughter.]

MR. CARROLL: At most.

MR. JOYCE: The reason for that is because when we do our review -- when we walk into an audit we are not fully equipped. We don't have what I will call the years and numbers of engineering effort that is put into a design. We show up on site or at the vendor's for a week and sometimes

two weeks and it depends on the complexity of the system, we may end up doing four or five audits.

Historically when we do show up, we do bring a multi-discipline team of staff members and consultants to go in. We pick a few subjects, and we go in and do a thorough review in that area of which we are doing an audit. If you said let's take the same team and let's quiz them about the rest of the design, we would fall short.

MR. LEWIS: I'm not going to argue the case, but not because I agree with you but because I also have a responsibility for keeping us on schedule.

[Slide.]

MR. JOYCE: Type of upgrades. This has to do with the present systems that we are looking at today. Since this last reorganization we have been looking at a number of retrofits that are going back into the plant, retrofit being that a utility is taking out a piece of equipment that is worn out or needs to be replaced, or decides to put microprocessor in for some other reasons.

The type of upgrades that we have been seeing is a direct replacement of a single analog function with a digital equivalent. That would be like the one that we looked at, Palisades where they put in a thermal margin monitor combining several analog process steps into a single microprocessor. That would be similar to the one we did up

at Haddam Neck where they are taking angates and orgates and set points and putting them into a single microprocessor.

Partial replacement of an analog system with digital. Partial replacement that Diablo Canyon put in a signal median selector that was right in the middle of a system. It wasn't the whole system, it was just partial replacement. Complete replacement of an analog system with a digital system. That was Prairie Island that put in a complete digital feedwater control system. From sensor all the way down to the control we had a complete digital system.

MR. CARROLL: Just as an aside, how has that worked?

MR. JOYCE: In terms of our --

MR. CARROLL: Has the system been very satisfactory?

MR. JOYCE: The feedwater control -- John Gallagher can answer that.

MR. GALLAGHER: I just left Westinghouse, and I would say that to the best of our knowledge talking to the customer he has been very satisfied with its performance, both with respect to its reading the requirements and the operators have been very satisfied with it. A lot of effort went into that to also deal with the man-machine interface.

There were some small problems with the way that

the AMSAC system was hooked in. I think they have been straightened out now.

MR. CARROLL: Thank you, John.

MR. JOYCE: Addition of a digital system that interfaces with the plant, that would be plant safety monitoring system. We saw that at Beaver Valley and a couple of other places that put in a plant safety monitoring system. This last one is Arkansas, where they replaced the core protection calculator. The first slide that I had which was a minicomputer, they already upgraded their microprocessor.

The next set of slides, because of time, I am not going to go into any detail. They are there for your information. What they are is, they are going to show you the present designs that we are looking at that has the plant's name, what the vendor is. For example, South Texas put in a QDPS, Qualified Display Processing System. It was built by Westinghouse. We reviewed it and wrote a safety evaluation report in May, 1987. The QDPS is a system that does a little protection, does some control, and it does some class I-E displays.

The main thing that I want to focus on for the next three slides -- like I said, I am not going to put them up there -- they are there for your information. You can see that we go all the way up to current, it is current. I

guess the last one shows that this month we issued an SER on Turkey Point where they are using Eagle 21 for upgrading their RTD bypass manifold.

[Slide.]

The main thing that I want to focus on in this slide is that the review process and the technique, tools, and criteria that were used for evaluation for these systems to date --

MR. KERR: Excuse me. What is an RTD bypass?

MR. JOYCE: It is where they take the resistant temperature detector. It is in the manifold. They take the RTD's out of the manifold and bypass around the manifold, and they are using the Eagle 21 -- I didn't review this system --

MR. KERR: Was it in effect, a replacement of the RTD by another system?

MR. JOYCE: Yes. Like I was saying, the three things that I wanted to focus in on three slides were with respect to criteria. The criteria are basically the same criteria that I showed to you on an earlier slide called lessons learned. We also recognize that verification and validation is not the only tool that can generate quality software. This standard should be augmented and updated to reflect some of the tools and methodologies that exist today in the industry and that have been proven.

With that, last year we sent over to research -- you heard from Leo Beltracchi right before me -- we sent over to research a thing that we call a users need that had approximately 14 items on it. This is just a short list of some of the things that we asked research for their help. It certainly is not all encompassing.

[Slide.]

We said in 279 we need a digital standard similar to like our IEEE standard 279 that talks about things like data communications. Data communications came up this morning with viruses, one way communications to transmit only, no hand shaking, security, reliability, diversities. These were all the subjects that we put into this user need.

MR. MICHELSON: Excuse me. What is Firmware?

MR. JOYCE: Firmware is when it is stuck into the hardware. It is soft --

MR. MICHELSON: When you take it --

MR. JOYCE: Like concrete when it sits up.

MR. MICHELSON: No. I don't know if it is that. Are these cards that you would plug into the other pieces of equipment like in a breaker; is that what you mean by Firmware?

MR. LEWIS: That would be a PROM is a standard example.

MR. MICHELSON: Okay, the PROM is a Firmware.

MR. LEWIS: It can be a whole card.

MR. WILKINS: It could be a chip these days, couldn't it?

MR. MICHELSON: I don't know.

MR. WILKINS: You can do an awful lot of things with one chip.

MR. MICHELSON: Yes.

MR. LEWIS: Yes.

MR. MICHELSON: It's firm, in a sense that it is non-programmable and so forth after it is burned in.

MR. LEWIS: The fact is that programmable is irrelevant --

MR. MICHELSON: It is. Not hardware -- I was thinking of that as hardware.

MR. CARROLL: This list that you are showing us here is actually from the January 25th letter from Gillespie to Beckjord that we have in front of us?

MR. JOYCE: Yes. That is one of them. Actually, there were three that were generated. Back on April 26th we wrote the first one, which is a mirror image of that one. Then in December, Dr. Murley sent one over to Beckjord that had the same ingredients in it. Research got it three different times through different channels.

MR. CARROLL: If I just read the January 25th one I know everything that is in the other two letters?

MR. JOYCE: Yes, and then some.

MR. SHEWMON: Why was it so necessary to write three separate letters?

MR. WILKINS: You think it was because they ignored the first two letters?

MR. JOYCE: No. Gosh, I don't want to do that. We wrote the memo saying we need help. We documented it and it goes up the chain, and then there's other programs going on where they are trying to pull other things together for research and prioritize their work. I don't know how it got lost -- I don't know how to answer that other than it is there now.

MR. CARROLL: It would seem to me that an awful lot of these things were things you needed or should have anticipated needing five years ago. Why didn't you ask research --

MR. LEWIS: Only five?

MR. WILKINS: I was going to say ten.

MR. CARROLL: Why didn't you ask research for all this laundry list way back when?

MR. JOYCE: That's a fair question.

MR. CARROLL: I only ask that kind.

MR. KERR: You don't really want an answer to that. What you want him to do is do something that he can do something about, and he can't do anything about that.

MR. CARROLL: I would just like to know the history of this thing because it troubles me that the Agency, it seems to me, is behind the ball game here.

MR. KERR: It's a different agency now than it was five years ago, so you aren't going to be able to learn anything useful.

MR. CARROLL: All right.

MR. LEWIS: At the risk of --

MR. CATTON: I would still like to hear the answer, even if what Bill says is true.

MR. JOYCE: All right. It's interesting in the sense that when we got V&V there was a competence level by the staff with respect to verification and validation. We have performed a number of audits, and the audits that we have performed that Ray Ets will talk about in the review methodology did prove -- these audits using verification and validation methodology did prove out to flush up out of the design errors and techniques -- not necessarily procedural errors -- software errors. The vendors that are here can stand up and speak to it say that's not necessarily true.

What happened is, we got a competence level with verification and validation and felt some what secure about it, just like we did with 279. You give me 279 in a plant and I feel great. Let's go and see what we can find, let's do some single failures and separations and seismic on them.

MR. CARROLL: It wasn't that easy with 279 20 years ago when it was invented.

MR. JOYCE: That's right.

MR. CARROLL: It was a nightmare to everybody.

MR. JOYCE: What we are doing is, we are breathing easy a little bit because we had such a burden prior to this with the earlier designs. You saw the man years we spent. We got V&V. We are still developing and it's an ongoing thing, we have people involved in the standards. Jim will talk about how we are going to upgrade standards.

There are lists. I can go back and pull out memos about here's another hit list that somebody ought to help us with. Why does it not get to research -- I probably shouldn't even answer that. We were thinking about it and it never got there. Like Dr. Kerr said, we are not going to gain --

MR. KERR: We have wallowed in enough nostalgia. Let's get on with it.

MR. LEWIS: Let me ask you two questions, both of them rhetorical. The first one is, are you going to issue some letter to the community telling them what V&V means, so they don't all come in here and mean different things by it?

MR. JOYCE: It's issued.

MR. LEWIS: The second question, because that one doesn't require an answer --

MR. JOYCE: Let me answer the first one.

MR. LEWIS: No. Let me go on to the second. The question is, in your first viewgraph you listed four speakers in this hour and you are the first one -- I assume that you are managing the time.

MR. JOYCE: Yes.

MR. LEWIS: Now you can answer the first one.

MR. JOYCE: Right there. Are we going to issue anything to the world --

MR. LEWIS: Does that define V&V?

MR. JOYCE: That defines this, and this defines V&V.

MR. LEWIS: That defines V&V?

MR. JOYCE: Yes.

MR. LEWIS: Why don't people read the definition and use it? I don't know what definition is in --

MR. JOYCE: V&V -- everybody has been talking about verification and validation all day today, and nobody was disagreeing with the terms. I ran off and made a slide, and that's what we mean by V&V.

[Slide.]

That V&V is consistent with 493, it's consistent with Westinghouse's definition, it is consistent with Combustion's definition that was given this morning.

MR. LEWIS: What is this page from? Is this a

page from our Reg Guide 1. --

MR. JOYCE: I copied it out of 7-4.3.2, definitions. Someone can check that for me, and I will check it when I sit down.

MR. LEWIS: Okay, thank you.

MR. JOYCE: That is what we have been using, that's the statute.

MR. LEWIS: That is your definition, okay. I just wanted a definition.

MR. JOYCE: Like I said, it is consistent with everybody up here that talked about it, except maybe yours was flip flopping back and forth a little bit.

MR. LEWIS: I don't think so, but go on. I will read that more carefully.

MR. JOYCE: Does Combustion agree with this? Does Westinghouse agree with this?

MR. MICHELSON: It's hard to read it.

MR. CARROLL: Why don't we make a copy of it so that we can read it.

MR. LEWIS: Let's make a copy so we will all know what we are talking about.

MR. JOYCE: With respect to time, I showed you the research. The next person we are going to have speaking is Jim Stewart. Jim Stewart is going to get up and talk a little bit about the future designs, what he is seeing

today, evolutionary plant or passive plant, and even the retrofits that I have touched on a little bit.

MR. STEWART: My name is Jim Stewart, with the I&C Branch. As a quick aside to Mr. Carroll's comment, my computer science program has a required statistics course in it.

[Laughter.]

MR. MICHELSON: That makes him a statistician.

[Slide.]

MR. STEWART: I put up this slide yesterday. This is just to show the plants that we are looking at, going from the ones we are actively and currently involved with down through very conceptual level that we don't have much detailed information on. So far, the vendors have told us that the passive plants in the I&C area will be pretty much the same philosophy of design as the plants are currently looking at.

MR. KERR: This is future applications of --

MR. STEWART: These are plants that we have not, as of yet, --

MR. KERR: I am looking at the title. The title is future applications of something.

MR. STEWART: Future applications of our review. It's probably not a great title.

MR. CARROLL: Of our review of digital --

MR. STEWART: Of digital systems within these plants.

MR. KERR: It's future applications of digital systems.

MR. STEWART: Right. We left retrofits and upgrades on the bottom there, simply because we do expect more retrofits and upgrades similar to what Mr. Joyce talked about before.

[Slide.]

Design features. When we made this slide we didn't have the benefit of knowing what Combustion and GE and EPRI were going to say. They have addressed all of that. The only one I would address in addition is expert in AI systems down at the bottom. The reason why I want to specifically mention that is that even though none of the current plants that we are looking at are intending on using it, they are all being fairly careful to leave the option open for possible future use. I will address --

MR. CATTON: What you call the DNBR, is that --

MR. STEWART: I believe it's very, very difficult to draw the line between what we have as computers now and what you call an expert system.

MR. KERR: AI is a buzz word that people use to get research contracts, Ivan.

MR. CATTON: I understand that.

MR. CARROLL: Some plants are actually using such systems in non-safety related --

MR. STEWART: In non-safety applications, and I expect you will see mor of that before we see it in safety applications.

MR. LEWIS: There have, in fact, been some spectacular successes with expert systems.

MR. STEWART: We are not ruling it out. We haven't taken a position that it is not a possible thing that can be done.

[Slide.]

There has been quite a bit of talk about what our review criteria is and what it is going to be for the ALWR's. The first thing obviously is the standard review plan. It does not specifically address digital systems, it does not have a lot of useful guidance as far as details of what to look at in digital systems. It has a lot of good things as far as the need for quality and the need to assess the quality of the systems.

MR. CATTON: Do you require that the applicant meet those standards?

MR. STEWART: The standard review plan, yes.

MR. CATTON: All the ones that you have listed there?

MR. STEWART: I will talk about that.

MR. MICHELSON: The standard review plan, I think I hear you say is admittedly inadequate.

MR. STEWART: Yes.

MR. MICHELSON: What else does the potential vendor have to do besides meet the standard review plan? Obviously, it is not adequate, or at least that's your position.

MR. STEWART: That is my position.

MR. MICHELSON: So, how does he know what he has to meet and how do we document that?

MR. STEWART: One thing that we have done is, IEEE 7-4.3.2 which is a description of verification and validation, including the definition of what those words mean.

MR. MICHELSON: That's for software?

MR. STEWART: That's for software. We endorse that with the Reg Guide, which was issued to everybody.

MR. MICHELSON: I am back to hardware --

MR. CARROLL: That Reg Guide is dated --

MR. STEWART: The Reg Guide is 1985. The standard is 1982. I currently am on the working group to upgrade 7-4.3.2, IEEE working group. We have both computer science, masters, Ph.D. people and nuclear engineers on the team. We have the vendors, we have the NRC, we have academia, a wide variety of people helping out on that.

MR. MICHELSON: What is your schedule?

MR. STEWART: We have a draft being reviewed within the review process right now.

MR. MICHELSON: What is your schedule for getting out a revised standard review plan?

MR. STEWART: Unfortunately, I don't have a lot of control over INPEC's voting time. We are hoping to put it up for a vote within the year.

MR. KERR: Mr. Stewart, are you listing things that -- I take it this is sort of the background material that a reviewer uses?

MR. STEWART: Yes.

MR. KERR: Is this for an ordinary intelligent engineer, if one could find anybody like that, or is it anticipated that the person who does the review will have to be an expert on computers in some fashion? Are you designing this for in-house use by NRC staff or for a contractor?

MR. STEWART: For all the people involved. For example, what we are putting into the upgrade are requirements that I believe any engineer could understand. We are putting in requirements that a V&V program must be in place and must be used. There was some question earlier with one of the vendors -- I forget who -- who had up there 1012 and 880 -- IEEE 1012 and IEC 880, as examples of

verification and validation plans which they used to develop their own in-house standards.

We are currently trying to use 880 and 1012 in the 7-4.3.2 upgrade. The 7-4.3. upgrade is intended to address all computer applications in nuclear power systems, safety grade. We agree with everything that the previous speakers have said. We do not separate the hardware and the software. We think it all needs to be addressed. The revised 7-4.3.2 will be an attempt to do that.

The details of how you would take the requirements that are in the 7-4-3-2 -- assuming eventually that we get it endorsed with the Reg Guide -- the details of how to do a design, which computer language you should use, the details of the best ways to use that computer language are not part of that --

MR. KERR: I am asking for the details of the review process and, specifically, what sort of person would you expect could carry out this review process? Would he have to be a computer expert or --

MR. STEWART: The computer person within the NRC would be myself or a person like me.

MR. KERR: That's all I wanted to know.

MR. STEWART: Part of the process, and am important part of the process will be for all of the utilities and vendors to know what the guidelines are ahead

of time. We do think that 1012 and 880 are good V&V plans, and that's why we are trying to endorse them. They are very proscriptive, and they don't necessarily apply to all situations. Therefore, we do believe that it is appropriate for designers to take elements from these and blend them into an in-house standard that they can specifically apply to a design.

MR. KERR: Would you anticipate that a vendor, if you go to that, would submit a topical report indicating how he developed this and you would review that? How would you use that, in other words?

MR. STEWART: They could either submit a topical report saying here's our V&V plan and we could review that and say that it's okay by reviewing topical report. Currently the way most of them are being done is, when a retrofit or design certification submittal is given to us and then we will get it.

MR. CARROLL: You are open to the idea of approving a V&V plan for a vendor that he could use time and again through a topical report review.

MR. STEWART: Sure. Currently how that is happening is, we will issue an SER for a specific application and then somebody else will come down the road and use that same equipment with the same V&V at their plant, and they will just reference that they have used it

before. It simplifies our review.

MR. CARROLL: All right.

MR. STEWART: I think that's pretty much all I had to say on that. I did want to mention that in addition to my participation on 7-4-3.2 and then eventually they will pull NRC's participation when it gets to the Reg Guide, John Gallagher is the Chairman of one of the Subcommittee's under 880 also on our staff. We are very definitely involved with the standards. We are using them, they are not hard criteria that I can say this is an NRC regulation.

[Slide.]

These are review issues that are coming out of the ALWR reviews. Some of them we have pretty good answers for, some of them we don't. We are asking help for research. I will try to go through them as quick as I can here.

Diversity, one of the people asked how the vendors know what our positions are. It is because we go and meet with them and tell them. All of the vendors that we are currently reviewing I think have a pretty good idea of what we are looking at.

MR. MICHELSON: Are those positions relative to electronic types of controls that we are talking about now? Are those positions in the standard review plan or somewhere else that someone can read them, on the diversity?

MR. STEWART: The diversity issue I am going to

talk about here is not in the standard review plan.

MR. MICHELSON: Is it anywhere else one can read to determine what your position might be?

MR. JOYCE: We could probably start with NUREG-0493, defense-in-depth and diversity of integrated protection. There we talked about definitions of diversity, functional diversity, so we do some definitions.

MR. MICHELSON: You are saying that there is nothing unique about solid state control system, digital control systems that would change this diversity argument at all?

MR. STEWART: Yes, there is some unique features on software.

MR. MICHELSON: Where do I find your modification of your position back in 0493?

MR. JOYCE: You won't find the modification with respect to software --

MR. MICHELSON: No, I'm talking about hardware.

MR. JOYCE: Hardware, you will find it back in there. What we did is, from block concept address common mode failure. That technique went off and the common mode failures of certain blocks -- you look to see what was left.

MR. MICHELSON: You are saying you are still using the requirements of 0493 --

MR. JOYCE: That is what was on ---

MR. MICHELSON: -- in diversity area even today.

MR. JOYCE: That question goes out on every single application, and we have some commitments.

MR. MICHELSON: Is that in the standard review plan that tells the reviewer that's the position and to go back and use it?

MR. STEWART: The 0493 is not in the standard review plan. We agree, the standard review plan --

MR. MICHELSON: 0493 was written a long time ago.

MR. STEWART: We agree the standard review plan needs to be updated in many areas.

MR. KERR: Diversity is independently of how reliable the system is.

MR. STEWART: What we are going to tell you now is our concern in this area and what our intentions are. This is not a criteria that the NRC has endorsed with the Reg Guide. IEC 880, for example, does have some discussion of diversity, and we are trying as much as we can to stay in the same kinds of definitions. Our main concern is common mode failure of systems which use software for this particular item.

The particular concern that we have is a design error that has not been found and the possibility for that design error to take out all four channels at the same time.

MR. KERR: This is diversity in software, you

said?

MR. CATTON: Hardware.

MR. STEWART: I will get to that, okay? The best answer that we have come up with to date of how to address that common mode problem because we do not think you can prove that the software is 100 percent accurate or 100 percent reliable, because we don't think that can be proven. The best answer we have come up with to date is diversity. There is very many types of diversity: diverse programmers, diverse languages, CANDU reactor, goes all the way up to systems, languages, programmers, verifiers. There is many different methods.

We don't have a set position on which one of those various methods is the answer. All the vendors that we are looking at now have a different answer. They all have some level of diversity, but they are all doing it a different way and addressing it with different criteria. Every one that we have looked so far has a different application.

I don't believe there is a consensus in the industry of the best way to do it, and we go along with that. We do have questions over at research to help us in this area. We do believe that some level of diversity is required. That's our current position.

MR. LEWIS: We went through a lot of this conversation long ago, and I remember the example I used at

the time which was the requirement in twin engine airplanes to have a propeller engine on one wing and a jet engine on the other wing for diversity. Everyone laughs, but I really don't see that diversity in itself is a virtue.

MR. STEWART: Diversity may or may not be a virtue. I do believe it's a viable solution to common mode software problems.

MR. LEWIS: I don't agree.

MR. STEWART: We disagree.

MR. LEWIS: But this is a subject that we have discussed before, not you and I. I won't do it on my time.

MR. STEWART: We bring this out now. I know it's not a published criteria. We wanted to --

MR. KERR: You believe this in spite of having looked carefully at the other problems that diversity may introduce?

MR. STEWART: Like I said, there are many different kinds of diversity.

MR. KERR: No, I was saying the other problems that diversity may introduce.

MR. JOYCE: Like you talked about on the ATWS issues?

MR. KERR: On the ATWS issue and other issues.

MR. JOYCE: The answer is yes to that.

MR. STEWART: We are also even considering the

cost to the vendors of how to address it. Diverse software introduces a tremendous cost burden. That may not be the best way to do it.

MR. KERR: I thought you were requiring it not only of software but of hardware as well. You are not requiring it of --

MR. STEWART: That's one of the possible ways of achieving diversity. Trigger research reactors with the new General Atomics Control Council -- which are very simple reactors and don't have he complications of the massive decay heat removal problems, have an analog protection system, hard wired copper analog system, and they have a digital microprocessor software base protection system.

MR. CARROLL: Propeller engine and jet engine.

MR. STEWART: They both work.

MR. CATTON: And, some of them are in the basement of a university.

MR. STEWART: Some of them are in basements of universities. I think you are about four or five miles from one right here. It can be done. There are different ways of doing it, and for the sake of time I think we should leave it there and keep going on.

EMI, we talked about yesterday. It's an ongoing developmental area. We are in the process of trying to collect which of the criteria we think should be applied the

most. Expert/AI systems, right now if somebody came in and said they were putting a AI system in a safety function, we don't have review criteria. We would have heartburn on how to do it. It's a research issue right now. I will leave it with that. Nobody has said they are doing that, by the way.

Design certification level of detail, we talked about that a little bit yesterday. In the area of software, the basic question would be at what point in the process do we go and audit the software.

MR. MICHELSON: I thought the question was how much detail do you need for certification?

MR. STEWART: Yes. How much do I need before certification and how much can I do after certification.

MR. MICHELSON: Do you have any thoughts presently on that point?

MR. STEWART: I have many thoughts on the subject --

MR. JOYCE: Excuse me.

MR. MICHELSON: Any positions, maybe would be a better question.

MR. JOYCE: The Commission --

MR. STEWART: Don't worry, Joe, I am not going to get into it.

MR. MICHELSON: I know what the Commission thinks. I was just wondering if you had any thoughts. We will share

them later, no doubt.

MR. STEWART: We have had many discussions --

MR. MICHELSON: We will share them later.

MR. STEWART: -- we have many other groups working on it, okay? Passive plant criteria I talked about, and we are looking at that. Most of that will come out of the systems requirements. We talked about HVAC. Commercial dedication is simply -- I believe Combustion talked about it. When you use previously existing software there is ways that you can convince yourself that it's good software without necessarily doing the V&V effort yourself.

MR. MICHELSON: It seems to me that an evolutionary plant also has HVAC questions about protecting the solid state equipment and --

MR. STEWART: Yes, the specific --

MR. MICHELSON: -- you list that. Is that because it's conventional or something, and this something more --

MR. STEWART: The HVAC at the evolutionary plants is redundant safety grade channels with safety grade backup power. MR. MICHELSON: Yes, but it all has to be ventilated and has to be kept cool. We don't even have standard review plans for chiller systems, for instance, which is predominantly what are being used.

MR. STEWART: We do have in the standard review plan the requirement that the designer demonstrate that the

equipment is qualified for its environment.

MR. MICHELSON: That doesn't quite --

MR. STEWART: We do audit that testing and we do audit the installation.

MR. MICHELSON: Generally a standard review plan has a little more guidance to the reviewer than that, but if you think that's all you need --

MR. STEWART: We gave examples yesterday of where I have gone out and done those audits and found problems with it.

MR. MICHELSON: We are talking now about the evolutionary plants in the same context as the passive plants.

MR. STEWART: The passive plant issue --

MR. MICHELSON: No, I am talking about the evolutionary HVAC.

MR. STEWART: The evolutionary HVAC --

MR. MICHELSON: You don't have a standard review plan. It isn't a plan that exists, and you can't go out and look at it. You have to look at it on paper and have to review the paper, and what kind of a review procedure do you use to look at paper on chilled water systems --

MR. STEWART: As far as design certification?

MR. MICHELSON: Sure.

MR. STEWART: They have committed to provide

safety grade HVAC and keep the electronic within their design envelope.

Segmentation, I think you have heard enough discussions on what segmentation means from the vendors. We think segmentation is a good idea. I personally think segmentation is a good idea, and will go from there. Separation and independence, what I am talking about here is different from the traditional IEEE 279. What I am talking about here is the data intercommunications was brought up. We think it is important enough. In the new 7-4.3.2 we are attempting not only to put in words about one way dedication, we are trying to put in pictures of exactly what we mean which buffers can talk to which buffers so that it can be easily understood by all people.

MR. JOYCE: Jim, excuse me. You are going to have to -- because I now have the clock, you are going to have to move. We have another 20 minutes.

MR. STEWART: That hits through -- you can read the last couple of bullets there. I am pretty much at the end anyway, Jim.

[Slide.]

The last issue we had, we have talked lots about standards development. I don't think I need to add anything to that. We have been having a lot of technical exchanges with foreign countries and list a few of them here. We have

gone there, they have come here. We have taken advantage of the lessons that we have learned from Canada and France. John Gallagher will talk a little bit more about the EDF.

As the research develops and additional guidance we will take advantage of it, and we will put that into the review guides. Are there any questions?

[No response.]

MR. KERR: Are there questions?

[No response.]

MR. ETS: One of the advantages of being last is that everything has been talked about, so hopefully I can go through it real quickly. My name is Ray Ets, and I am a consultant to the ISCB, the Instrumentation Branch One of the things that we have been addressing here all day is just the software which now implements functions which previously had been done by the logic and analog devices.

Today, what I wanted to focus in on and what I have been asked to focus in on is two things; an overview of the criteria at the working level that has been used for the audits; and, secondly, some of the techniques that the NRC staff has used to implement these criteria.

MR. LEWIS: Just out of curiosity, could you tell us what Smartware Associates is?

MR. ETS: Smartware Associates is the trade name of my consulting company.

MR. LEWIS: I see, so you are Smartware Associates?

MR. ETS: Yes. As my wife calls it, smart ass, for short.

[Laughter.]

[Slide.]

MR. ETS: Basically what we have here is a key problem with regard to software that there is no consensus of what constitutes good software. For example, just today we have seen one approach using off the shelf modules and another approach using straight line code, another approach using a table driven set of modules. This is a problem in evaluating software and the review of that.

The NRC has determined or has made a decision to use verification and validation as the tool with which to evaluate the viability of software for use in safety systems. A key area here is that verification and validation, everyone has talked about it, but this is a separate and parallel process to the development of the software.

With regard to definitions, the verification is determining whether the process -- whether the requirements at one phase of the process have been completely consistently carried over into the next level of the development process, basically providing us a good feeling

or ensuring that there is functional correctness. The second part, the validation, this is a test of the integrated system. This is the hardware/software integration. To see whether it satisfies the initial specification at the highest level, the functional spec or system spec -- what the validation test does is provide a good feeling or high level -- I know it's ambiguous but a feeling of confidence that the software in combination with the hardware, the safety system, will in fact perform its safety function.

MR. LEWIS: Does that mean that in particular -- I hate to harp on definitions, but we have to get them straight. That means that you also are not using the strictly formal definitions of verification and validation that are common in the computer science community.

MR. ETS: The definitions that we are using are based on 7-4.3.2. We are using that as a basis, and those definitions are consistent with the definitions that are presented in IEEE 1012 which is the software community standard on verification and validation.

MR. LEWIS: When we speak of the functional representation or the functions that are supposed to be represented by the software in validating them, validating a function space is a matter of many inputs to many outputs, and real validations means that you check all possible

inputs against all possible outputs and make sure that the software matches what the original functional definitions -- is that overly restrictive?

You spoke in terms of having a good feeling, and I can tell you lots of ways I can get a good feeling but I can't put them on tape.

MR. ETS: That was the objective in developing a sufficient level of confidence that from the NRC point of view you can license this software for use in a safety system.

MR. LEWIS: In other words, you do not check the whole function space but just enough to feel good about it? I know what I am --

MR. ETS: Let me clarify the point. Again, the NRC as Joe alluded to, because of limited resources does not do the V&V function. What we are looking for is from the vendors point of view, it has the vendor applied the verification and validation process consistently in the development of the software. We would audit that, as I get to later, in taking a representative sample and taking a close look at that.

MR. LEWIS: I am just trying to get away from, if you will forgive me, from the excuses. I know the resources are limited. I am trying to get at the definitions. We are not talking about validating the full input space against

the full output space, which what in computer science is called verification and validation. You are doing it selectively. That's all right. There's nothing wrong with doing it selectively.

If you do it selectively, then there has to be an informed selectivity in picking that part of the space that you do validate. That's what you are going to talk about?

MR. ETS: I will be talking about how the NRC approached it, yes.

MR. LEWIS: Not quite the same thing, but I will wait to hear it.

[Slide.]

MR. ETS: The criteria for the safety software, being a regulatory agency, we have to fall back on the criteria that allows us to say that this is required and needed. You have the ANSI 7-4.3.2 which has been endorsed by Reg Guide 1.152. That has been discussed. We also look in at, in doing our audits in a real sense, we also use the guidelines of 0493 to look at the software system susceptibility to the common mode failures. We have actually done and put together block diagrams of the proposed systems and used that as the basis of our analysis for the common mode failures.

We have not neglected the other criteria that we use to provide general guidance and guidelines in IEEE 1012

and IEC 880, the IEC being the European standard for software in nuclear systems.

MR. LEWIS: Incidentally, since Jay hasn't asked you, let me be the heavy on this one. I am taking for granted that you are a really honest to golly computer scientist. With a name like Smartware Associates you must be.

MR. ETS: I do have a degree in computer science. I worked under John Carr at the University of Pennsylvania. I did my thesis under Noah. I hope that satisfies you.

MR. LEWIS: I took it for granted, and thought I would put it on the record.

MR. ETS: When we are doing an audit on a proposed software system these are the general subject areas that we focus in on. Number one is the process for V&V. Everyone has said that while we need validation and verification -- here we are specifically looking at does a V&V plant exist and does that plant include the procedures, how are you doing the V&V. We are looking at what the vendor is proposing or has done or has accomplished with regard to V&V.

The other key element is that the V&V is independent, as I stressed before. This is a process which occurs in parallel. It was brought out in GE's slide very well, in parallel with the development effort. Here with

regard to independence is the minimum criteria that the staff has looked for is that the first line supervisor for the verification team is different than the first line supervisor for the development, that they are not all under the same hat.

The third point is the application of V&V. They have a V&V plan and within the plan they have defined independence, but have they done the validation and verification throughout the software development process. A key thing there is that the V&V has to be formally documented. A very positive note today is that everyone is talking about applying V&V at the highest requirements level. In some of the audit reviews that we did in the last three years that has not always been the case from the point of view of the vendors.

[Slide.]

Looking at the requirements documentation, software -- this whole area of software is a documentation problem. Software as such doesn't exist. It exists as documentation, be it a requirements document or software design document, a flow chart, a program design language or the code itself. What is key is that all of the development process is fully documented, starting off with the requirements document.

Configuration management, although not strictly

stressed, configuration management is a key element. This shows the feedback mechanism that the vendor has in place so that when he uses the classical waterfall software development paradigm that is always presented -- you have the functional requirement, design test, integrate, et cetera.

You find errors in there. How do you feed back those errors, those lessons learned into the previous levels and do it in a way that is manageable and coherent. This is configuration management.

Finally, it is the developmental methodology they use. This really comes down to -- as required by 7-4.3.2 -- do they have the phases, the development phases fully defined. What is the product of each phase, how is that phase represented and through what documentation. Again, how are the errors handled. How are the errors handled when the product -- when the software or the integrated product is within the design groups somebody finds an error, are they tracked. Also, how are the errors handled when the errors are uncovered by the verification team. That is, after the product of that particular phase has been put under configuration management.

These are all the areas that we focus in on when we go to do the audit and get the information that we need to be able to find out whether the software can be licensed

and detail of design.

MR. LEWIS: I couldn't help but notice that you keep using the word we, but you are a consultant to NRC. Presumably you mean they.

MR. ETS: We, in the sense that I have been part of the audit team for the past three years. I use the we, meaning myself and the staff in doing these audits. There was a discussion previously about interdisciplinary teams and how you bring them together. I am a computer scientist and have had to learn a lot about nuclear engineering in the last three years. Hopefully, I have taught them a little bit about software engineering.

MR. LEWIS: Hopefully.

[Slide.]

MR. ETS: Basically on the audit methodology, as I stressed before, we are doing the vendors -- looking at the vendor V&V program, the pragmatic of the situation that we can't do it ourselves. The first point we do is, we have a series of questions that will look at -- when we get the document, we do a series of questions, the answers to which we will look at. The standard set has been developed by dissecting 7-4.3.2 which currently is our only enforceable criteria and seeing how many of those questions are answered in the initial documentation. What is not answered there would be presented as RAI's for the next go around in order

to get the response.

The questions are particularly important when we are talking about the advanced ALWR designs for which actual software or even a software design for that matter, has not been completed or defined. In the case of retrofit systems where an actual system exists and the software has been written, we use what is called a thread concept.

[Slide.]

On the thread concept what we do is, this is a schematic of a typical safety system with the sensor input. You have a conversion calculation block and a trip logic leading to trips. They trip signal out of one or four channels. What we do is, we select a thread starting from a sensor and going through these blocks all the way to the trip. The selection of this thread, perhaps in answer a little bit to Dr. Lewis' question of what inputs are we looking at, the selection of a thread is a team decision in which you have the electrical engineers who have the nuclear scientists and computer scientists -- we want a thread that will be as representative and as encompassing as possible with regard to the vendor's development of the safety system.

Each appropriate discipline then takes their section of the thread and takes a closer look at it. In the present day most commonly the software is used in the

conversion and calculation block, and that's the area where I have focused my efforts.

[Slide.]

Looking at the documentation of a supposedly simplified diagram, you find that the documentation peels back the cover of a seemingly simple conversion block and exposes the underlying complexity of the software itself. Very often designs are purported to be simplified, and they may be graphically simplified true by having what was previously a logic shown in an one liner replaced by little blocks saying that this is the calculation computer or this is a bi-stable computer or whatever.

What is really happening is that previously visible complexity has in fact been pushed down into that graphically neat block, and it is something that we cannot really forget about. There is a lot of complexity hiding in the software program as well as complexity added. The focus is not on whether we do mathematical functions in the software but rather how the software gets the data in, what it uses as a criteria for its various branches because that's the essence of the logic.

We focus in and get a view of what the software is like in the block. What we do is, select one of the modules in there from the block and take a look at that module's life cycle development as represented by the life cycle

document starting from, again, functional requirements, software design, looking at the code where possible, then looking at their results of verification. The document of verification that they did as the software module moved through the various phases of the validation, and also looking at the validation test of the completed subsystem.

Basically, looking at this slice will give us a good idea or impression of whether they were thorough in their design, whether the design covers all the aspects, whether the design elements have been v[illegible], whether the validation included exhaustive testing or random testing, what did they use as their test implement. Basically, again, it reduces to selecting one representative piece and examining it thoroughly.

[Slide.]

I am going to jump to my conclusions here. Basically what we have is that V&V is in fact a proven technique. It does uncover errors that otherwise would go uncovered, just for the sake of having somebody else look at it, another team look at it. Outside of the nuclear community I think a very prominent example is the Hubble Space Telescope's mirror in which it was said that independent reviewers can't be used on this because nobody knows this subject as well as we do.

Apparently that argument was bought, but we see

what the results are.

MR. LEWIS: That was a result of not being able to measure lengths to within a millimeter, and it was partly the kind of smugness you describe. It was also partly a matter of having trouble feeding earlier tests which did show the mistake up through the layers of management so somebody noticed it.

MR. ETS: The other half of it, yes, there was very precisely ground -- it was the correct answer to the wrong question, would be another way of putting it.

MR. LEWIS: Since you mentioned the Hubble Telescope I will say one thing. I just read the report on it a few weeks ago, and the report told me that they used the well known Hindle Sphere test to test the telescope. I asked all my astronomer friends who had never heard of it, I went and looked at my library and found one 1931 book which mentioned it. It's not all that well known.

MR. ETS: That happens. Basically, it's a proven technique. The other thing is, V&V is technology independent. Although in our case we are using requirements, we have the list of questions and a list of database. That is not to say that the application of V&V cannot employ computer-based tools. This would be especially true if, for example, a vendor has a well integrated case development facility from which it would be

much more -- we provide much more insight to the NRC reviewer of which tests were done and would lead one to a much more confidence that the software as well as the system had been validated and verified.

Finally, as I said, the 7-4.3.2 is the handle that we have had to use in doing the reviews. I think that when I first saw it I personally expressed an opinion that it was kind of weak, but that's what we have to work on. We have been using throughout the reviews -- we have been using the other standards, and in particular 1012 and IEC 880 as guidelines in how better to assess and evaluate the system software of safety systems.

Basically, one of the things that I wanted to stress again that I skipped over before, standards -- we keep on coming back to that - standards, in fact, reflect a certain body of accepted opinion within a discipline. Standards are evolutionary. The software engineering is, in fact, an immature discipline compared to some of the others. It is maturing, and I think in evidence of that is looking at the amount of new standards that have been propagated by the IEEE in the last four years.

I think this is where the NRC should look for additional help and assistance to having their standards evolve as the consensus of opinion within the software community as to what constitutes good and reliable software.

Thank you.

MR. LEWIS: Thank you. I have two questions. One is, would you feel more comfortable if there were another real, live computer scientist working with you?

MR. ETS: To be honest with you, yes.

MR. LEWIS: Second question. I am just curious whether you know from your involvement in the trade whether, for example, the 767 which is a highly computerized airplane that is built by Boeing, whether they did any formal V&V during the design of the airplane. I just don't know the answer to that question.

MR. ETS: I haven't looked at it specifically, but I understand that Boeing does have a good software development program. Whether how independent the V&V was, I don't know. I can't answer that.

MR. LEWIS: I know they have very good people. I wonder if they went through formal V&V on the system.

MR. ETS: I think they had to, to get FAA certification. FAA does have a requirement on the avionic system that is similar to what the DOD requires, where in fact Boeing would have had to have an indeperdent -- the FAA would have an independent contractor team to do the verification and validation in the ultimate sense of independence, really.

MR. LEWIS: Thank you very much.

MR. CARROLL: I have one question. We have heard this morning the Canadian experience.

MR. ETS: Yes, sir.

MR. CARROLL: I guess I came away with the impression that the licensing authority in Canada and their consultant or consultants had a real hard time accepting the software development and V&V that went into the Darlington design.

Would you have the same problems that were described this morning if you were looking at that design in Canada? I just want to get some calibration as to whether they are being more rigorous that we are.

MR. ETS: I am not familiar with the V&V that the CANDU went through, so I can't answer that.

MR. KERR: My impression is that the goal of the staff as this point is to ascertain that the licensee, applicant or whatever have a reasonable V&V program and that it has been applied in a reasonable way but not to attempt to estimate the reliability that results therefrom. Is that a reasonable conclusion?

MR. ETS: You asked about the staff, and I think perhaps the staff should answer it. What I do is, I give my own report and that's input to the staff report. They can take my impressions with a grain of salt, so --

MR. STEWART: Are you talking about placing a

reliability number on the software?

MR. KERR: A number, or some other indication of reliability.

MR. STEWART: We have looked at what some of the vendors have done. I don't believe that there is a consensus in the industry of a way to measure a reliability number for the software. We have it as a research question, to see if that can be done. As of right now, we don't think that there's a way to do that.

MR. KERR: I am used to this sort of thing because in the education field what we do is, we have a process and have no idea what the results of that process ultimately are, but we spend a lot of time worrying about the process. I guess that is sort of an analog here.

You have faith that the process will produce something that is your goal but you have no way of really measuring that.

MR. STEWART: We are using verification and validation because we believe it's a proven process that gives us a high level -- granted, we can't put a number on it -- a high level of confidence that the software is of good quality and will perform its function as intended.

MR. KERR: Thank you.

MR. GALLAGHER: I would just like to say one thing before I start with respect to the international standards.

I am the Chairman of what is called Subcommittee 45-A which is responsible for writing the standards on systems and equipment for the use in nuclear reactors. It is equivalent to the IEEE group that writes these standards, but it is on an international basis. The U.S. is a voting member of this international basis.

We work on a lot more than just 880 and are now in the process of revising 880. It was written in the 1980's, and we realize that it did not address the use of a lot of software tools and more modern methodologies that are now available. I would say that two-thirds of the people who do this actual writing are computer scientists. I am not sure exactly what their degrees are but that's what their work is, and they represent the software expertise in their various countries.

MR. CARROLL: John, it might be interesting for the Committee for you to tell them a little bit more about yourself. You are newly arrived at the NRC --

MR. GALLAGHER: Yes, I just joined the NRC in January the 7th. I spent my years, from 1956 until I left Westinghouse at the end of November, 1990 working in the area of the development of advanced I&C products for the Westinghouse plants, starting with the ion chamber, the protection systems that are in the operating plants, then on to digital technologies. I was the manager of the IIS

project that was spoken about earlier.

More recently, I worked in the application of digital technologies to the feedwater system. Most recently looking into networks and things like that to improve the instrumentation system capability with respect to incorporating decision making processes.

MR. CARROLL: I have known John for a long time, and I can say that I have had to make systems that he has developed work in the real world.

MR. LEWIS: Have you succeeded?

MR. CARROLL: Yes, we usually figured a way around the problems that he created.

[Laughter.]

MR. LEWIS: Are there any successes that you have had in apply neuro-networks to decision making systems of the kind that you are talking about?

MR. GALLAGHER: We were just getting started into this and we found some interesting things. The people that I was working with --

MR. LEWIS: In other words, the answer is no.

MR. GALLAGHER: No, the answer is yes, because the people that I was working with were doing this for the defense business and were able to see things on the ocean floor that nobody else could see. They were building and shipping neuro-network systems for this purpose, and they

worked.

MR. LEWIS: I know something about that stuff.

[Slide.]

MR. GALLAGHER: I would like to just give you a sort of overview of what the French N4 I&C system is, because I think there's a lot of confusion about what is being talked about. This is a view, starting up here at the top with the control room, you hear about Level three, two, one zero. Level three is the control room. Below that is the processing and communications system which takes in information from the plant and processes it, sends it up to the operators and also develops signals or commands that are then sent down.

Below this is what the level one. It basically is the control level. It is made up of the safety system which is the reactor protection system and the signals that go out to activate the engineered safety feature systems. This was developed by Framatone, and it basically -- when it is in operation it's in the 1,300 megawatt reactors and sustained fairly well.

The area that is being talked about is the P20 I&C system which covers this system. What they are talking about and the problem is how to fix this, and I will get into it a little bit later what the problem is of how to fix this and keep this, and especially keep this. It is also

interesting to note that there is a mimic ar gram and auxiliary panel that go directly down into th 20 system and bypass all the data.

[Slide.]

This is a view --

MR. KERR: You are going to tell us what the P20 system is supposed to do?

MR. GALLAGHER: Yes, I will. The P20 system, this is another view of the same thing. The difference on this view is that it shows that down at this level there is a large number of these controllers, the P20 type of controller. There is roughly 13 or so of them that communicate with each other as well as send information up.

[Slide.]

The control room has work stations. This is the -- it has four of these work stations, two for the operator. This is for the senior reactor operator, this is for the reactor operator, this is for the shift supervisor, this is in the technical center. This whole thing is driven by either central computers, the Gould machines, or by microprocessors including the computers that serve to operate the work stations.

The French spent in this area, the documents show that they spent somewhere on the order of 60 engineers for four years. That's 240 man-years of engineering. They

would like to keep this. They spent about $50 million on that, the problems in this area.

[Slide.]

What does the P20 look like and what does it do. It's a microprocessor based system that has a data highway. It has up at this end what they call the cluster head which serves to perform the functions and make the data highway operate. These functions are identified here. There is coupling into the plant networks with are basically ether networks. There is a coupling into the interim cluster, so the one cluster which is this group, can talk to another cluster. There is the management of the traffic, things that do calculations.

Here is the block that does the maintenance configuration and monitoring. Down at this level are the connections from the data highway out into local networks, local buses, where you pick up the measurements, the analog and digital measurements and perform the controls. This is a distributed digital processing system made up of microprocessors.

MR. KERR: I have to ask a stupid question. What is this supposed to do? What is it for?

MR. GALLAGHER: It is for the purpose of measuring process variables and performing control algorithms.

MR. KERR: So, it's a control system.

MR. GALLAGHER: That's right. It's basically a distributed digital processing control system with its own proprietary data highway and being able to hook into an open highway system.

The problem that they ran into -- let me just say one more thing about that. The problem they ran into appears to be three-fold. One of the problems is the large amount of data that is being processed, much larger than previously. If we look here there is some 65,000 digital signals that are being processed. If you look at prior experiences on the 1,300 megawatt reactor there were 5,000 digital signals. The earlier estimates on this job were something like 20,000.

They are a factor of three above their earlier estimates which they made around 1985. One of the things that we have heard is that they significantly increased the number of equipments that were being monitored. For instance, a lot of the valves that are normally manually operated and are entered into the -- if there is a computer system and they are entered manually, all that was done now by automatic monitoring.

There was a large increase there. Also, and I spoke to the man that wrote the document that put these numbers down. He is going back to check this. This now says that there is something like 200,000 points per second, so

when you add this up you end up with a total digital rate of something like 265,000 plates per second which is much larger than anything that I think of any us have ever seen anybody talk about. The analog is about normal.

[Slide.]

Evidently, this is their problem as it shows up. At a very high data rate there is evidence that they took advantage of the advanced computational capabilities within the microprocessors to work around this high data rate, which means they got very clever with their programming. Rather than using some of the rules that you heard people talk about earlier, very strict restrictions on how you modularize, they were working on ways where you could part data, how the place you were to go was filled up and where you could go elsewhere and things like that.

There was obviously, I think, uncertainty in the specifications. The system kept growing, so that as the development of the system was going along the specifications were being changed. There has been some evidence to that fact. The equipment was new. It was new in two ways. When the job originally started out they had planned on doing some of this with a different set of equipment. Over the course of time that company was acquired by somebody else, somebody else who now makes the P20 wanted to make everything the same product line so that was changed.

They made a change to what their original plans were, and they also introduced -- as best as I can find out, there is not a lot of experience in the application of this particular equipment. When you add these three things up, you end up with complex programs and poor documentation. Then you say okay, not let's go do and V&V program. You cannot do the type of V&V program that you heard about with complex programs and poor documentation.

I think something very similar to this happened last year on the Mohasha plant in Czechoslovakia. I don't know how many of you know, but there was a plant that was using a distributed digital system there. They had engineered it. As they went through the engineering process they kept changing the specifications and they had some of this problem. When they took it to Kiev where they were supposed to do the V&V program they ended up with the very same problem here.

I think that one of the lessons here is that while V&V certainly puts rigor into this, what is equally important is a design process that realizes that at the end you have to go through a V&V program. You heard some of the people who recognized this saying that one of the chief roles of the rules that the design process has to follow is to make sure that you end up with a product that you can do verification and validation on.

They are now in the process -- they, being the French -- they are in the process of figuring out what they can do, and hopefully make a decision within six months on how they can go to something else and still be able to save most of their control room product.

Are there any questions on this?

[No response.]

MR. GALLAGHER: Okay, thank you.

MR. LEWIS: Thank you very much. That finishes our formal set of presentations. Does the Subcommittee have any further questions for any of the speakers?

[No response.]

MR. LEWIS: In that case, I think we should probably relieve the transcriber of her duties. I don't know what the rule is. We want to have a little bit of conversation around the table. I don't think we need to transcribe it.

MR. ROTELLA: It's up to you.

MR. LEWIS: You are relieved. The formal session is over.

[Whereupon, at 4:43 p.m., the meeting concluded.]

REPORTER'S CERTIFICATE

This is to certify that the attached proceedings before the United States Nuclear Regulatory Commission

in the matter of:

NAME OF PROCEEDING: Joint Meeting -- Computer in Nuclear Plant/Control System

DOCKET NUMBER:

PLACE OF PROCEEDING: Bethesda, Maryland

were held as herein appears, and that this is the original transcript thereof for the file of the United States Nuclear Regulatory Commission taken by me and thereafter reduced to typewriting by me or under the direction of the court reporting company, and that the transcript is a true and accurate record of the foregoing proceedings.

Mary C. Lashin

Official Reporter
Ann Riley & Associates, Ltd.

USE OF COMPUTERS IN CANDU STATIONS

FOR

ACRS MEETING
BETHESDA, MARYLAND
91–02–06

N. ICHIYEN
(AECL–CANDU)

BRUCE A
FUELLING MACHINE INCIDENT
ACTIONS TAKEN

HARDWARE:

A number of hardware changes are being made/considered: for example,

- ac power to bridge, carriage and trolley drives are being moved to a non–bypassable bus
- protective output only a permissive, requires another command signal
- addition of hardwired interlock for bridge motion if head is moved forward

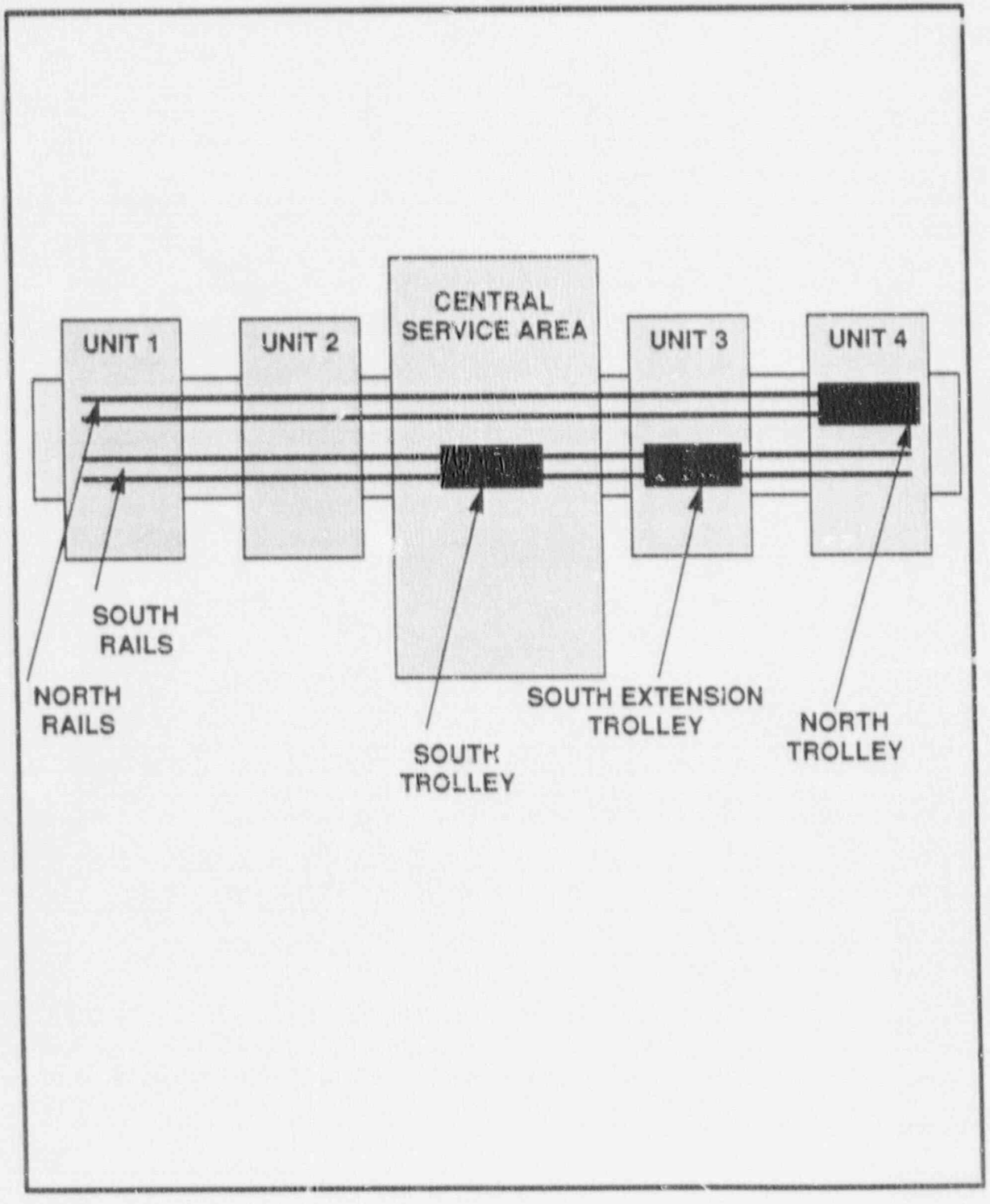
BRUCE A
FUELLING MACHINE INCIDENT
ACTIONS TAKEN

SOFTWARE:

A number of software changes are being implemented/considered: for example,

- "bug" fixed
- separate bridge motor and brake software
- hazard analysis completed
- SQA review and changes

BRUCE G.S. "A" FUELLLING SCHEME



BRUCE A
FUELLING MACHINE INCIDENT

EVENT

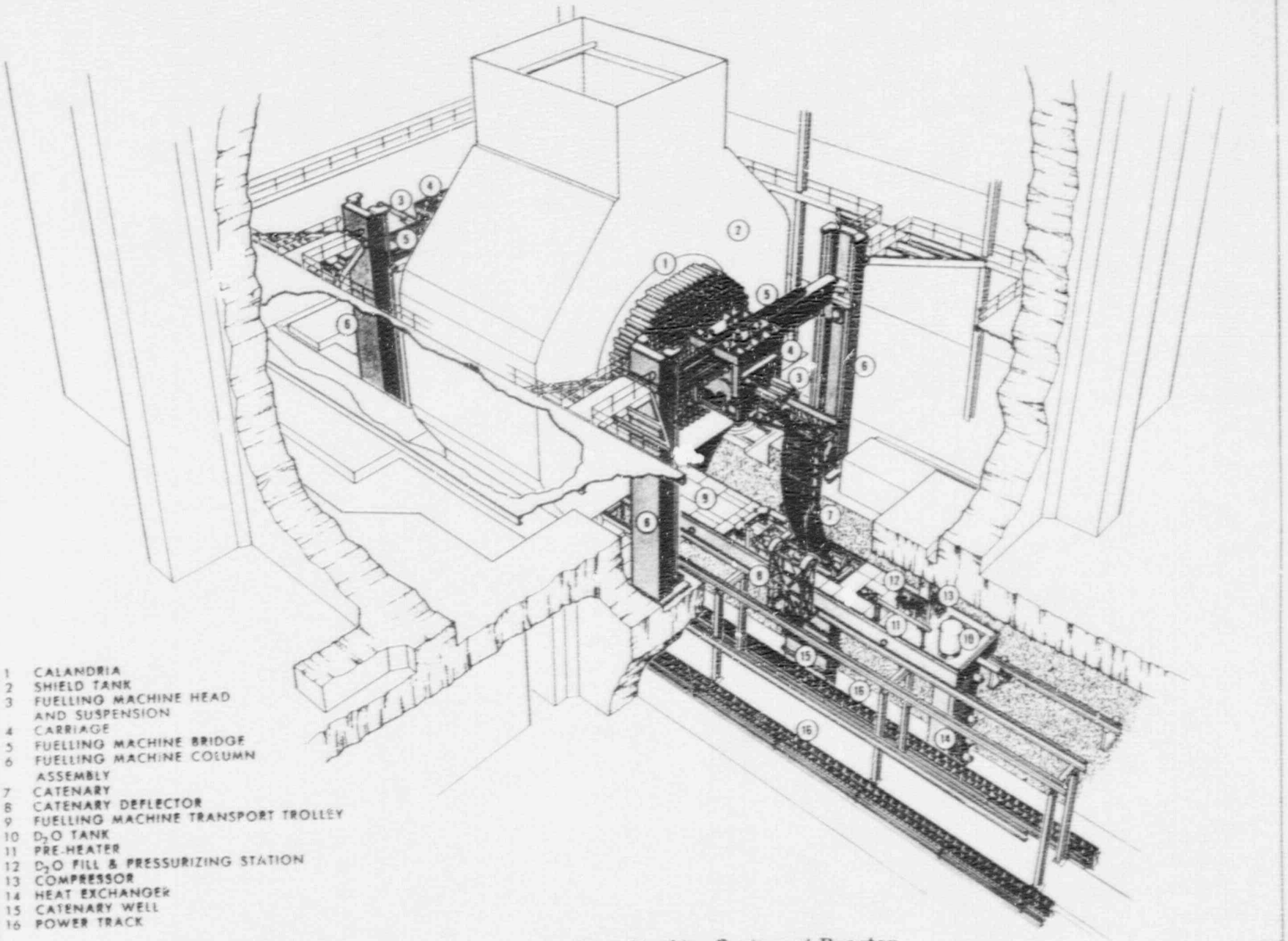
- the Operator was trying to carry out a manual control function (using the South Extension controls for the South trolley while it was in the CSA), due to an equipment failure (abnormal but normally permissible). Note use of manual control disables the AC protect circuit.

- this was not allowed by the protective computer (because the South trolley was in the CSA)

- this caused the protective computer program to enter a section of code that had a specific "bug" that caused the program to jump to a subroutine that ended up releasing the brakes to the Unit 4 fuelling machine (last access of the subroutine was from a Unit 4 operation to release the breaks)

- it so happened that the Unit 4 machine was actually latched onto a fuel channel and this resulted in the machine coasting down and causing a leak from that channel

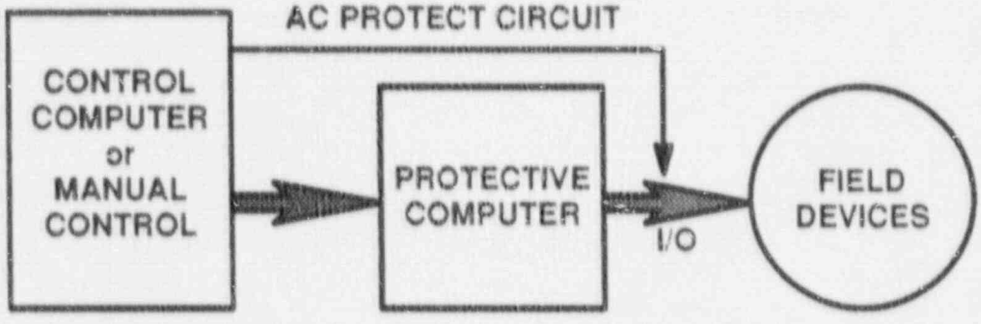
- manual reactor shutdown and cooldown proceeded without further incident



1 CALANDRIA
2 SHIELD TANK
3 FUELLING MACHINE HEAD AND SUSPENSION
4 CARRIAGE
5 FUELLING MACHINE BRIDGE
6 FUELLING MACHINE COLUMN ASSEMBLY
7 CATENARY
8 CATENARY DEFLECTOR
9 FUELLING MACHINE TRANSPORT TROLLEY
10 D_2O TANK
11 PRE-HEATER
12 D_2O FILL & PRESSURIZING STATION
13 COMPRESSOR
14 HEAT EXCHANGER
15 CATENARY WELL
16 POWER TRACK

Fuelling Machine System at Reactor

COMPUTER CONFIGURATION



OVERALL STATUS

- Safety critical high level standard just issued for trial use.
- Sub–tier standards, procedures, guidelines, to be completed by end of 1991.
- Decision on safety critical methodologies, configurations by mid–1991.
- Other category standards work also now underway.

BRUCE A
FUELLING MACHINE INCIDENT

BACKGROUND

- 4 unit station (4 x 848 MWe)
- on–power re–fuelling using computers
- 3 fuelling subsystems for the 4 units
  - each subsystem has 2 fuelling machine heads carried on a mobile trolley system
  - 2 sets of tracks for the trolley
  - once trolley is positioned at the selected unit, each fuelling machine is raised to the reactor face elevation by a carriage supported by a bridge structure

THEORY OF STATISTICALLY VALID RANDOM TESTING

Hardware reliability is defined as the probability that a failure occurs given a demand.

Software reliability can be defined as the probability that the software will encounter a demand which causes it to fail.

- distribution of inputs presented to the software (including time histories) must duplicate the real demand distribution (operating profile)

- an appropriate number of tests must be performed to gain the required statistical confidence.

FUNDAMENTAL PRINCIPLES OF HIGH LEVEL STANDARD FOR SAFETY CRITICAL SOFTWARE

5. Hazard Analysis

   - identify any failure modes that may lead to an unsafe action and thus either eliminate them or, where possible, ensure that the failure mode can be detected and the system put into a safe state.

FUNDAMENTAL PRINCIPLES OF HIGH LEVEL STANDARD FOR SAFETY CRITICAL SOFTWARE

4. Both systematic and random testing must be performed.

   - white box
   - black box
   - randomly generated (statistically valid random testing)

STATISTICALLY VALID RANDOM TESTING

- random testing is seen as complementary to systematic testing.
- provides added/independent confidence in the robustness, correctness, trustworthiness and reliability of the software.
  - improves effectiveness of testing by compensating for false assumptions and biases of the tester.
  - can be defined simply as testing using inputs selected at random from some known distribution.

FUNDAMENTAL PRINCIPLES OF HIGH LEVEL STANDARD FOR SAFETY CRITICAL SOFTWARE

2. The outputs from each development process must be reviewed to verify they comply with the requirements specified in the inputs to that process.

   - those outputs using mathematical functions must be systematically verified against the inputs using mathematical verification techniques or rigourous arguments of correctness.

FUNDAMENTAL PRINCIPLES OF HIGH LEVEL STANDARD FOR SAFETY CRITICAL SOFTWARE

3. Software structure must be based on "information hiding" concepts.

   - the interface to each software module is designed to reveal as little as possible about module's internal workings.

   - as a result, if it is necessary to change the functions internal to one module, the resulting propagation of changes to other modules is minimized (easier to maintain)

   - results in loosely coupled modules and hence is easier to review as well.

HIGH LEVEL STANDARDS

- defines requirements on the software engineering process
- defines outputs of that process
- defines requirements to be met by each output
- specified as measurable as possible but does not unnecessarily constrain the methodology to produce the output.

FUNDAMENTAL PRINCIPLES OF HIGH LEVEL STANDARD FOR SAFETY CRITICAL SOFTWARE

1. Documentation must describe the required behaviour of the software using mathematical functions written in a notation that has clearly defined syntax and semantics.

   - more complete requirements (domain coverage can be checked)
   - requirements can be uniquely interpreted
   - facilitates use of mathematical verification techniques that allow the design to be transformed into a mathematical function form for comparison to requirements directly.

REAL ISSUE

- lack of an accepted definition of the acceptable quality that the software had to have in order to be approved by the AECB.

- our objective is to create a set of standards, procedures and guidelines for software engineering over all categories of software.

    - our first task is the creation of this set for safety critical software.

STANDARDS FRAMEWORK

4 PARTS:

1. Categorization criteria
2. High level standard
3. Sets of standards, procedures, guidelines
4. Pre-developed software qualification

ISSUES FROM DARLINGTON A SHUTDOWN SYSTEM SOFTWARE LICENSING EXPERIENCE

- Drawn out licensing process
  - from 1985 (start of dialogue)
    to 1990 (approved for full power operation)
- Issues kept changing
- Different set of issues, from AECB–hired consultant, Dr. Parnas (1987)

ACTIONS TAKEN

1. Back engineering a software design specification using mathematical notation ("formal").

2. Walkthrough process to verify that the code met the formal software design specifications.

    - involved creating new techniques never used before like creating Program Function tables from the code and comparing to the formal specifications.

3. Random testing program.

CANDU 3
EVOLUTION OF DIGITAL SYSTEMS

Control

- from redundant "central" type system to a true distributed control system architecture
  - geographic distribution
  - closing the loop over the highway

Operator Interface

- separate PDS (Plant Display System) from control computers for operator interface

Safety Systems

- higher degree of computerization (i.e. more systems)
- evolutionary software practices

SOFTWARE ENGINEERING PROCESS

- concentrate on safety critical software category
- will describe the overall approach for producing reliable software
  - will discuss parts played by V&V, software reliability measures, etc.

DARLINGTON A COMPUTER SYSTEMS

DCCs

- reactor and process control (device logic control done in PLCs (OH–180s))
- operator interface (MCR)
- alarm annunciation
- data logging

Fuel Handling Control

- separate computers for on–line fuelling control

Safety Systems

- Fully computerized shutdown systems (SDS–1, and SDS–2)
  - trip functions
  - operator displays
  - operator aided testing
  - monitoring of important SDS variables
- Emergency Core Cooling System (ECCS)
  - use of PLCs (OH–180s) for discrete logic control

CANDU 3

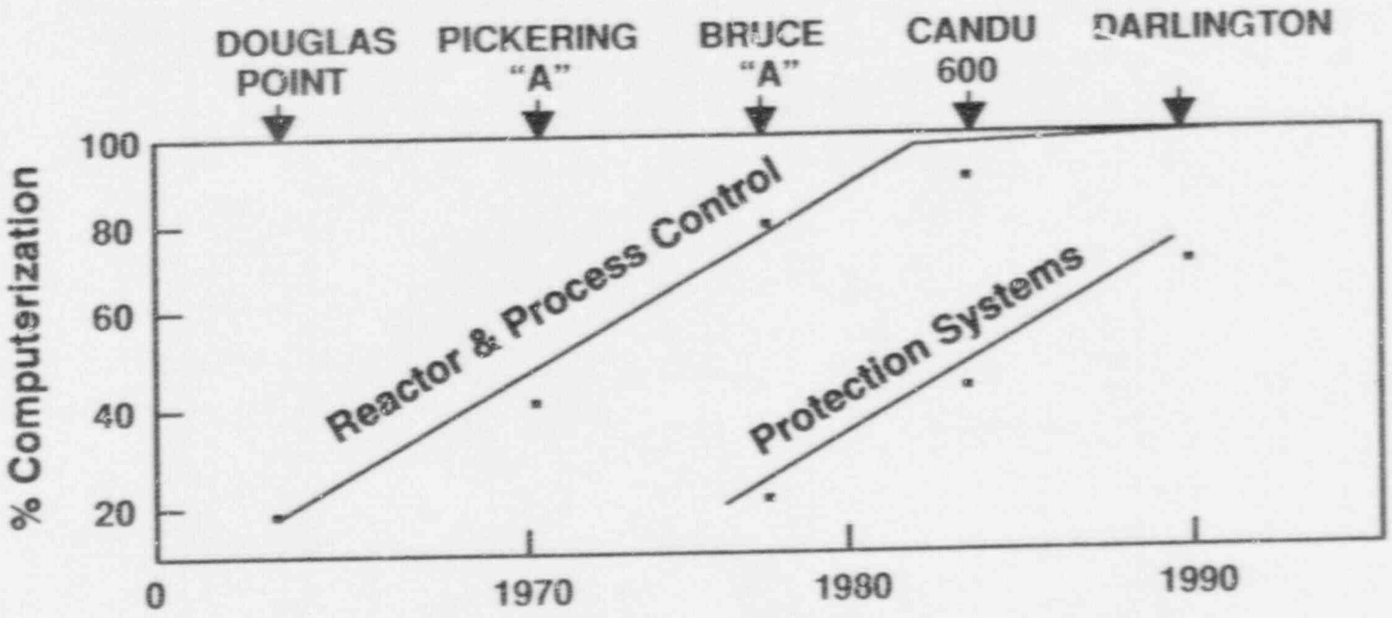
- Next generation CANDU after Darlington

- Features
  - 35 month construction schedule
  - modular design/contruction techniques
  - 100 year life
  - replaceable fuel channels
  - all equipment can be replaced within a 90 day outage (including st. gen, etc.)

OUTLINE

1. Background
   - historical
   - Darlington station
2. Future applications – Evolution Of Digital Systems
3. Software Engineering Process (concentrating on safety critical software)
   - Darlington shutdown system licensing experience
   - lessons learned
   - direction for future (use of standards)
4. Fundamental Principles of High Level Standard for Safety Critical Software
5. Overall Status
6. Bruce Fuelling Machine Incident

Accumulated Computer Experience in CANDU Power Plants



(2)

NUPLEX 80+

SOFTWARE RELIABILITY

KEN SCAROLA

MANAGER, ADVANCED CONTROL COMPLEX ENGINEERING

NUPLEX 80+ SOFTWARE RELIABILITY

- DETERMINISTIC DESIGNS
- FIELD PROVEN EXECUTIVE SOFTWARE
- SOFTWARE DESIGN PROCESS AND DOCUMENTATION
- SEGMENTATION
- DIVERSITY
- SABOTAGE PROTECTION
- EXPERIENCE

DETERMINISTIC DESIGNS

- INPUTS ARE SCANNED AND PROCESSED ON A CONTINUOUS CYCLE REGARDLESS OF STATUS CHANGE.

- SIMILARLY, OUTPUTS ARE UPDATED ON A CONTINUOUS CYCLE

- PROGRAMS ARE EXECUTED ON A CONTINUOUS BASIS

  NO MULTI-TASKING
  NO INTERRUPTS

- PROGRAMMABLE LOGIC CONTROLLERS IN SAFETY SYSTEMS EXECUTE PROGRAMS WITHOUT BRANCHING.

FIELD PROVEN EXECUTIVE SOFTWARE

- ALL SOFTWARE BASED SYSTEMS ARE COMPOSED OF COMMERCIALLY AVAILABLE PRODUCTS WITH PROVEN INDUSTRIAL AND UTILITY PERFORMANCE:

  - PROGRAMMABLE LOGIC CONTROLLERS
  - PC-AT COMPUTERS
  - MINI-COMPUTER
  - CRT WORKSTATIONS
  - ELECTRO-LUMINESCENT DISPLAY WORKSTATIONS
  - COPPER AND FIBER-OPTIC COMMUNICATION NETWORKS

- MOST OF THESE ARE USED IN NUCLEAR APPLICATIONS

- FIELD PROVEN EXECUTIVE SOFTWARE INCLUDES:

  I/O HANDLING
  ARITHMETIC FUNCTION BLOCKS
  COMMUNICATION DRIVERS
  FAILURE DETECTION

SOFTWARE DESIGN PROCESS AND DOCUMENTATION

- o EARLY FOCUS ON ESTABLISHING CORRECT REQUIREMENTS AND SPECIFICATIONS:
  - HARDWARE/SOFTWARE
  - FUNCTIONAL DECOMPOSITION
- o STANDARD CODING AND DOCUMENTATION TECHNIQUES
- o THOROUGH VERIFICATION AND VALIDATION PROGRAM
- o EXTENSIVE CONFIGURATION CONTROLS
  - PURCHASED SOFTWARE
  - CUSTOM SOFTWARE

NUPLEX 80+ SYSTEM DESIGN

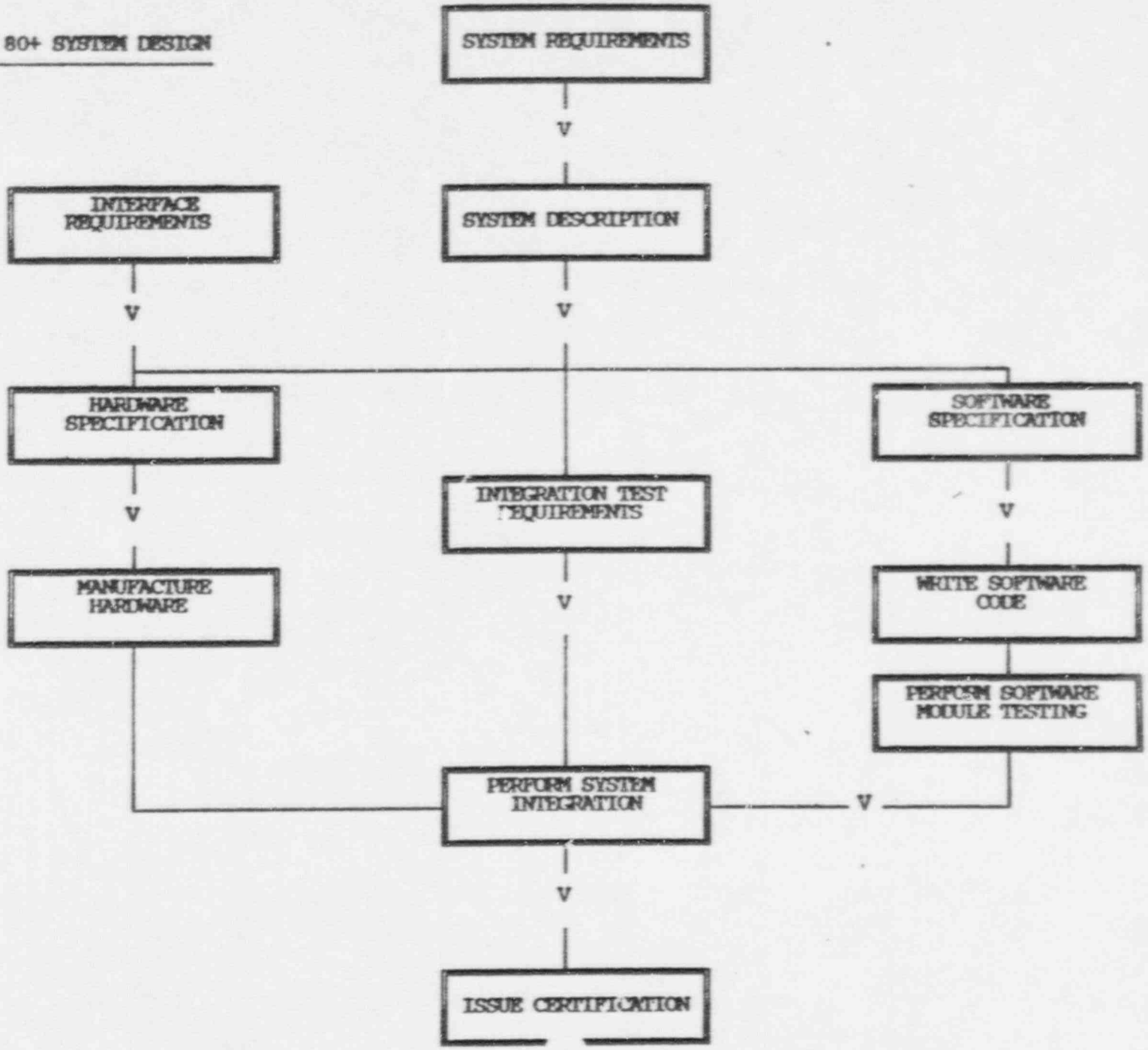


Figure 2-0
V&V Reviewers

	Non-Safety	Important to Safety or Availability	Safety
Functional Requirements	RT/DT or DT/VT	RT/DT or RT/VT	RT/DT&VT
System/Software Description	DT/RT or DT/VT	DT/RT or DT/VT	DT/RT&VT
System/Software Specifications	DT/DT	DT/RT or DT/VT	DT/VT
System/Software Implementation	DT/DT	DT/VT	DT/VT
Module Test Proc	DT/DT	DT/VT	DT/VT
Module Testing	DT/DT	DT/VT	DT/VT
System Test Procedure	DT/VT or DT/RT	DT/VT or DT/RT	RT/VT
System Testing	RT/DT or VT/DT	RT/DT or VT/DT	RT/VT

Key XX/YY

XX = Originator

YY = Reviewer

DT - Design Team

RT = Requirement Team

VT = V&V Team

Safety Sys: PPS, E-CCS, PAMI

Important Sys: DIAS, P-CCS, PCS

Non-Safety: DPS, NIMS, SOE

SEGMENTATION

- BREAKS SYSTEM FUNCTIONS INTO SMALLER UNITS EXECUTING ON SEPARATE PROCESSORS.
- ADDS A LEVEL OF DEFENSE AGAINST COMMON MODE FAILURES BY INTRODUCING:
  - FUNCTIONAL DIFFERENCES
  - CODING DIFFERENCES
  - EXECUTION TIME DIFFERENCES
  - HARDWARE DIFFERENCE
- PARTITIONS MORE PROBABLE FAILURES INTO MANAGEABLE UNITS

SYS80+ RT FUNCTION vs TRIP PROCESSOR ASSIGNMENT

TRANSIENTS \ TRIPS	SG1 lo P	SG2 lo P	CONT hi P	SG1 lo L	SG2 lo L	SG1 dP	SG2 dP	SG1 hi L	SG2 hi L	PZR lo P	PZR hi P	LOG PWR	DNBR lo	LPD hi	VOPT	CONT hihi
FW temp decrease	1*	2*											CPC*	CPC	1	
FW flow increase								1	2				CPC	CPC		
Main steam flow increase	1	2											CPC	CPC	1	
IOSGADV	1	2											CPC			
SLB i/o containment	1	2											CPC			
LOL											1,2					
YTRIP											1,2					
Loss of cond vacuum								1	2		1,2					
MSIV closure											1,2					
Loss of non-emerg AC to station aux													CPC			
Loss of norm FW flo				1	2						1,2					
Loss of RC flow													CPC			
1 RCP seizure													CPC			
RCP shaft break						1	2									
Uncont CEA withdraw at low pwr											1,2		CPC	CPC	1	
at power													CPC			
1 f/l CEA drop																
s/u of inactive RCP																
Core flow rate incr																
Inadvert deboration											1,2	2	CPC	CPC	1	
CEA ejection															1	
CVCS malfunction											1,2					
SG tube rupture													CPC			
LOCA										2			CPC			

1* - BISTABLE PROCESSOR 1
2* - BISTABLE PROCESSOR 2
CPC* - CORE PROTECTION CALCULATOR

TABLE 1

System 80+ RT Function vs Trip Processor Assignment

DIVERSITY

- OPERATING PLANTS EXHIBIT SIGNIFICANT DIVERSITY. NOT BY DESIGN, BUT RATHER BY THE RESULT OF ANALOG TECHNOLOGY AND CONTRACTING OF NUMEROUS SUPPLIERS.

- THIS EXCESS OF DIVERSITY MAY ACTUALLY DETRACT FROM PLANT SAFETY DUE TO:

  PERSONNEL TRAINING
  REPAIR TIMES
  SPACE PARTS AVAILABILITY

- NUPLEX 80+ MAXIMIZES STANDARDIZATION WHILE MAINTAINING A MINIMUM LEVEL OF DIVERSITY TO OFFER THE FINAL DEFENSE AGAINST COMMODE FAILURES

- DIVERSITY IS EMPLOYED WHERE SOFTWARE BASED COMPONENTS ARE UTILIZED.

NUPLEX 80+ DIVERSITY

- NUPLEX 80+ MAXIMIZES STANDARDIZATION WHILE MAINTAINING DIVERSITY IN KEY AREAS TO ENSURE THAT THE DEFENSE IN-DEPTH CONCEPT IS NOT COMPROMISED

- NUPLEX 80+ DIVERSITY:

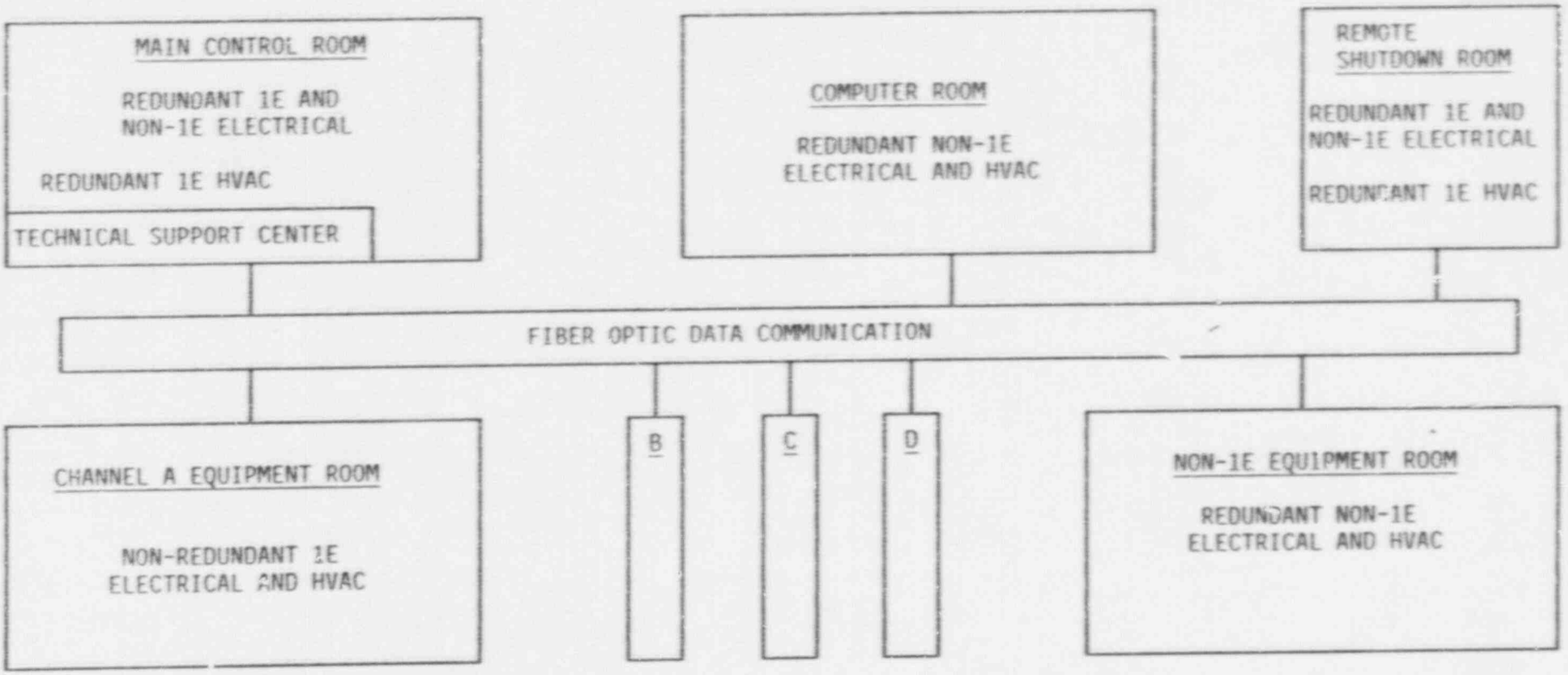
FUNCTION	DESIGN TYPE 1	DESIGN TYPE 2
REACTOR TRIP	PLANT PROTECTION SYSTEM	ALTERNATE REACTOR TRIP WITHIN PROCESS-CCS
FLUID SYSTEM CONTROLS	EMERGENCY SUCCESS PATHS (E.G., EMERGENCY FEEDWATER) VIA ESF-CCS	NORMAL SUCCESS PATHS (E.G., MAIN FEEDWATER) VIA PROCESS-CCS
REACTIVITY CONTROLS	EMERGENCY BORATION VIA ESF-CCS	NORMAL CEA CONTROL - VIA POWER CONTROL SYSTEM
ALARM AND INDICATION	ALARM TILES AND DISCRETE INDICATORS - VIA DIAS	CRT DISPLAYS - VIA DPS

SABOTAGE PROTECTION

- CONFIGURATION CONTROL DURING DESIGN, CONSTRUCTION AND OPERATION
- GEOGRAPHIC SEPARATION OF SAFETY CHANNELS
- ROOM AND EQUIPMENT ACCESS SECURITY/ALARMS
- CONTINUOUS PROGRAM MEMORY CHECKSUM REPORTING TO THE DATA PROCESSING SYSTEM

SLIDE169.00C

NUPLEX 80+ SEPARATION AND ISOLATION



EXPERIENCE

- ABB/C-E HAS BEEN DESIGNING SOFTWARE FOR SAFETY SYSTEMS SINCE MID-1970'S.

- THIS INCLUDES SOFTWARE FOR:

  CORE PROTECTION CALCULATORS
  SUBCOOLED MARGIN MONITOR
  QUALIFIED SAFETY PARAMETER DISPLAY SYSTEM
  INADEQUATE CORE COOLING MONITORING SYSTEM

- FOR CPC's THERE HAVE BEEN 826 SOFTWARE CHANGE REQUESTS SINCE INITIAL INSTALLATION AT ANO-2

- 99% ARE FUNCTIONAL DESIGN CHANGES, NOT SOFTWARE BUGS

- THERE HAVE BEEN NO FAILURE TO TRIP ERRORS

3

SOFTWARE FOR ADVANCED REACTORS

PRESENTATION TO THE ACRS JOINT SUBCOMMITTEES

ON

COMPUTERS IN NUCLEAR POWER PLANT OPERATIONS

AND

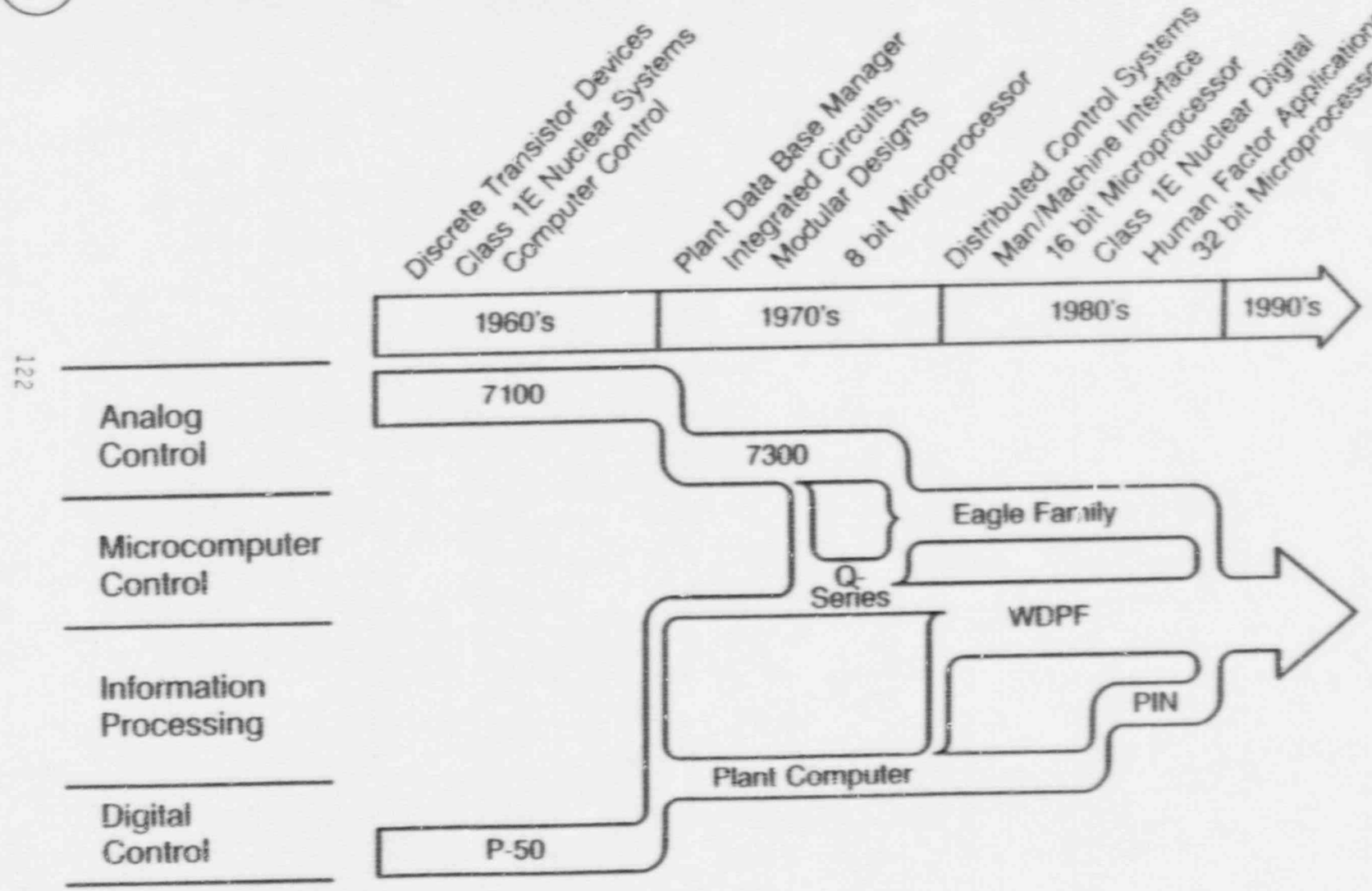
INSTRUMENTATION AND CONTROL SYSTEMS

FEBRUARY 6, 1991

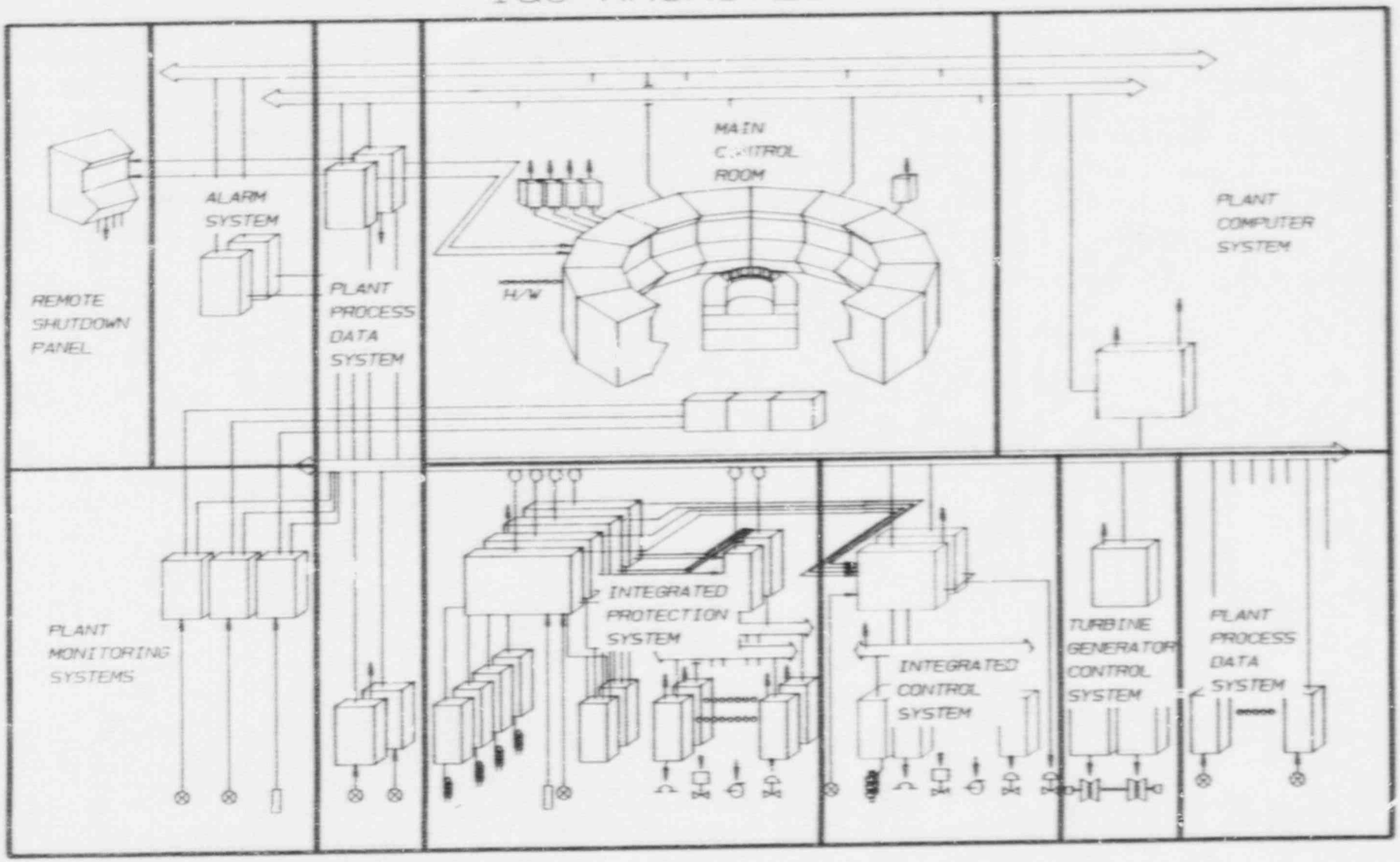
J. B. REID

OVERVIEW AND OBJECTIVES

Instrumentation & Computer Product Evolution



I&C ARCHITECTURE



OBJECTIVES OF

WESTINGHOUSE I&C SYSTEM DESIGNS

USE DIGITAL TECHNOLOGY TO PROVIDE IMPROVEMENTS IN:

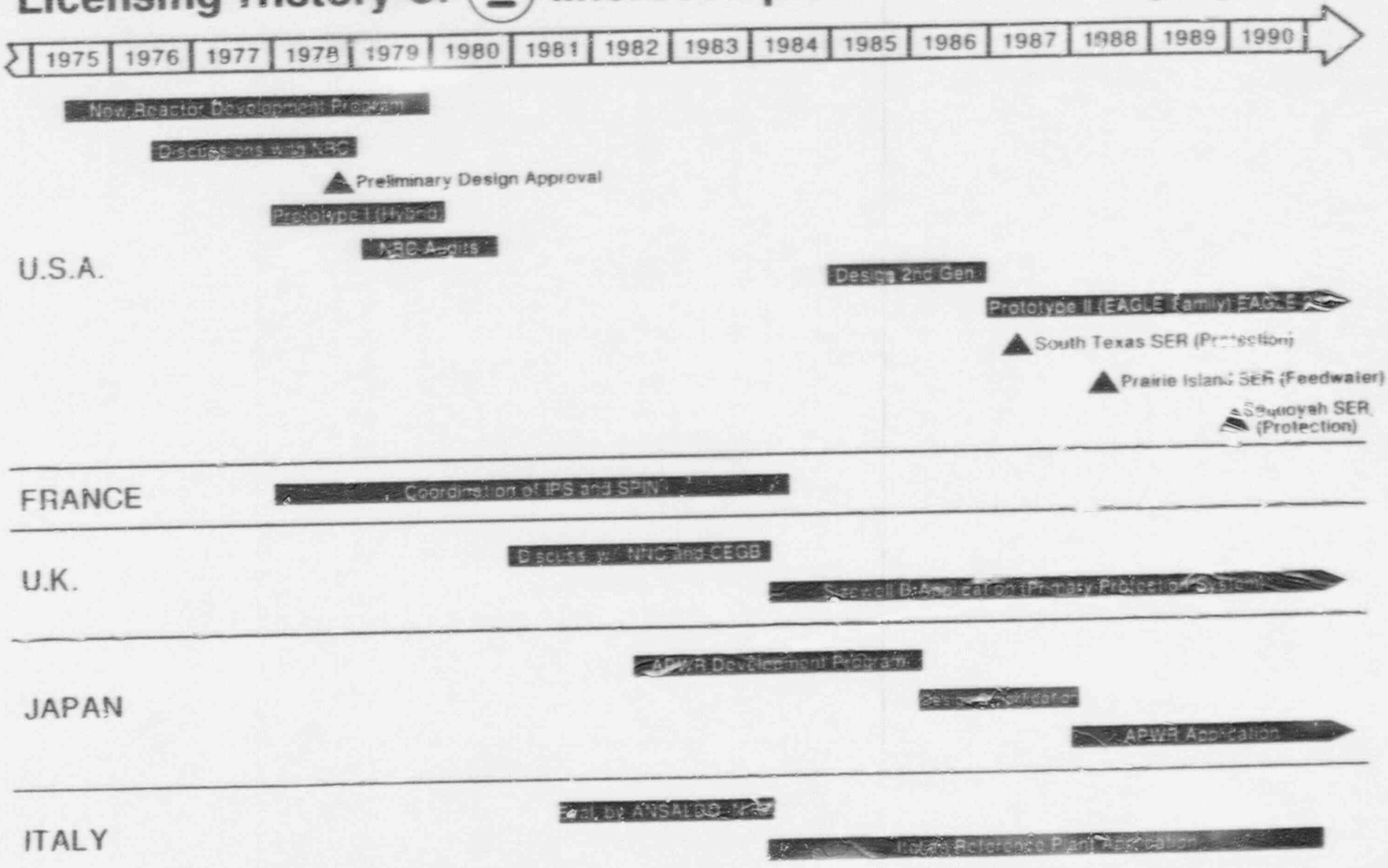
- COST
- SCHEDULE
- CONSTRUCTABILITY
- MAINTAINABILITY
- OPERABILITY
- FLEXIBILITY
- RELIABILITY
- LICENSEABILITY

INTEGRATE AND UNIFY THE TOTAL PLANT I&C SYSTEMS

I&C Architecture Characteristics

- Modular Design
- Digital
- High Performance where necessary
- Distributed Processing
- Data Highway and Data Link Communications
- Physically Distributable
- Hierarchial Architecture for Communication and Data Transfer
- Fiber Optic Cabling
- Fault-Tolerant Design
- Clean separation within safety equipment and between safety and non-safety equipment
- Improved Control and Protection Algorithms
- Information Presentation in Context with Navigational Aids

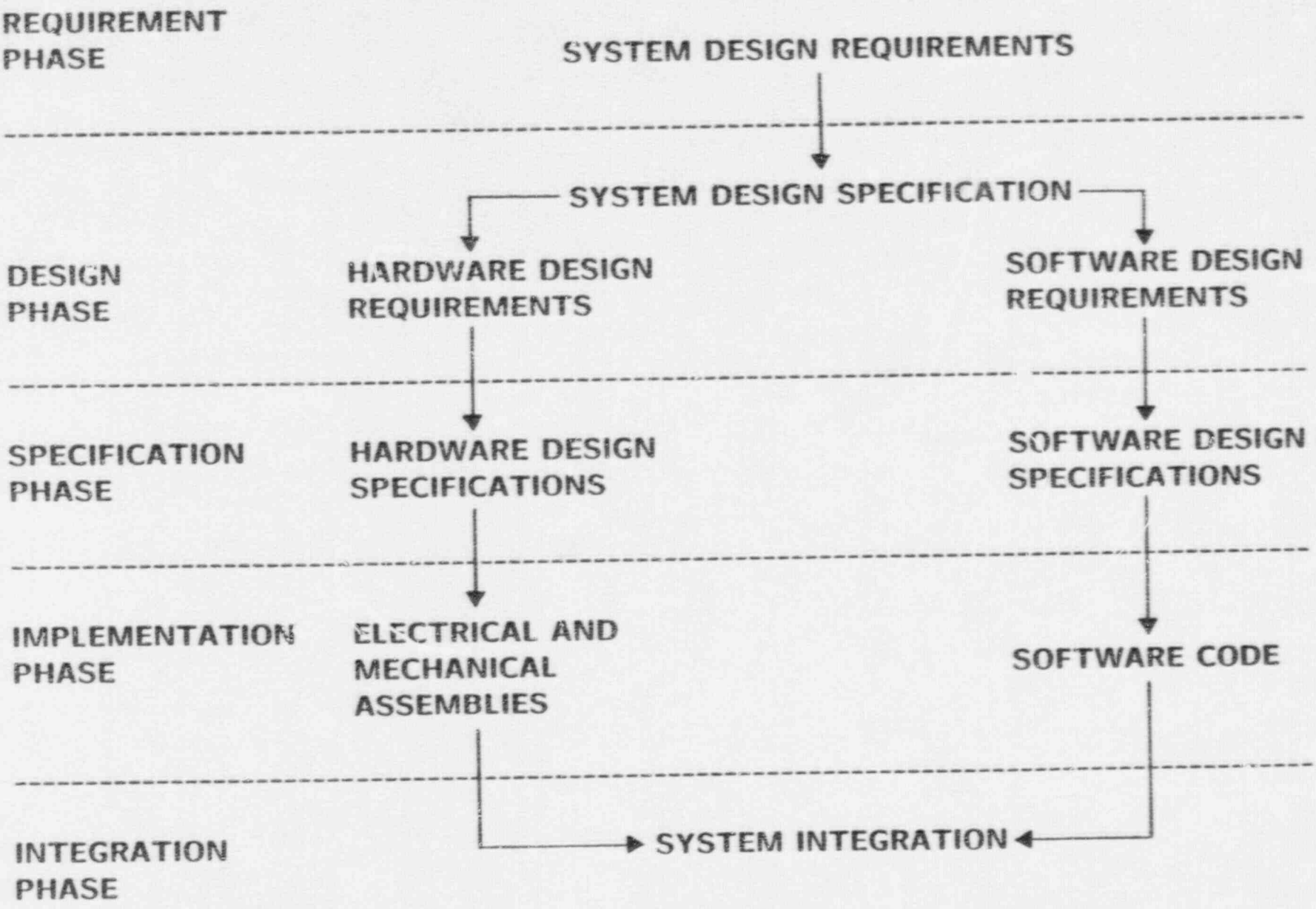
Licensing History of (W) Microcomputer Based Safety Systems



ZQ21.ZQB DISC 110

SOFTWARE DESIGN PROCESS

DESIGN PROCESS



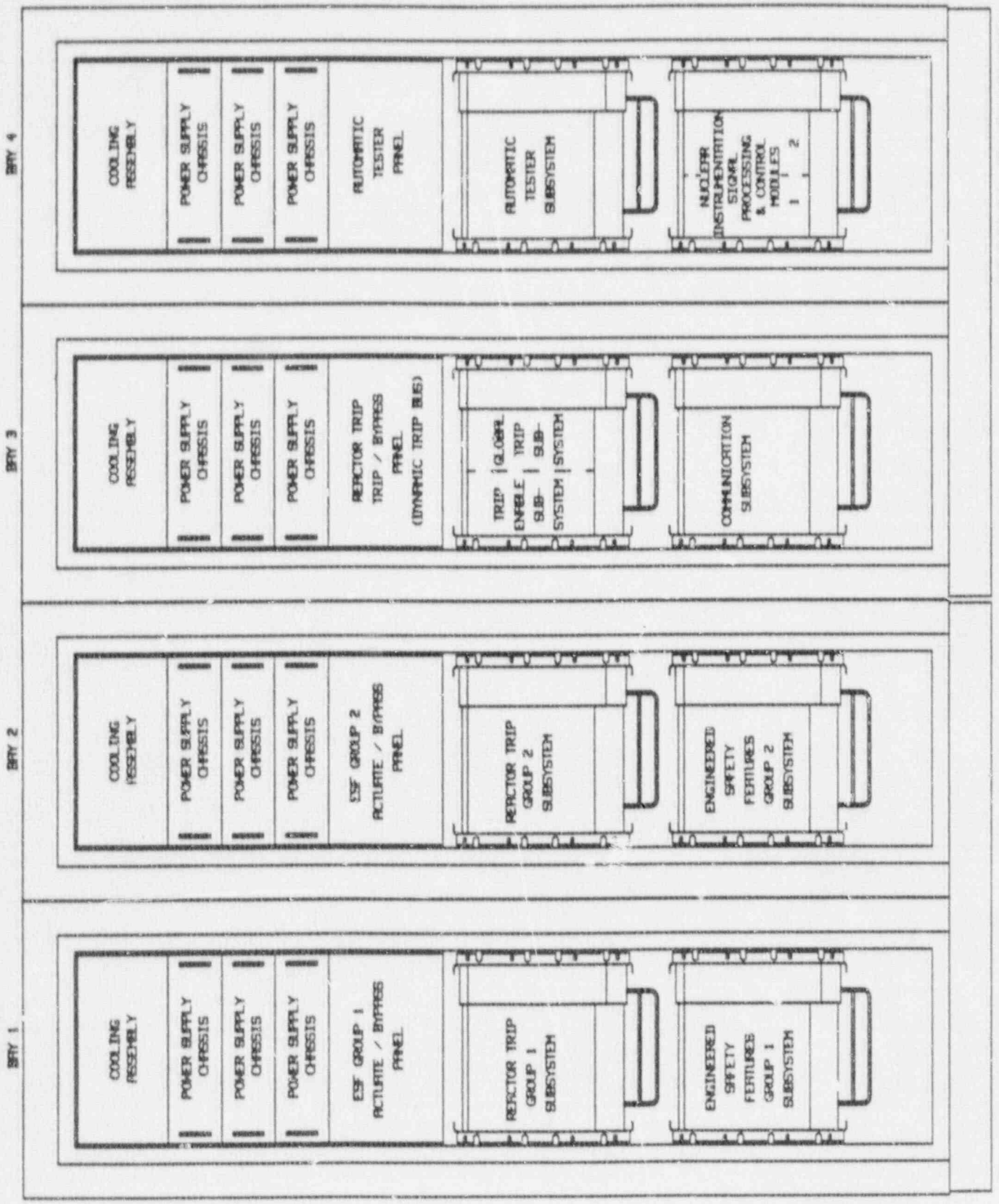
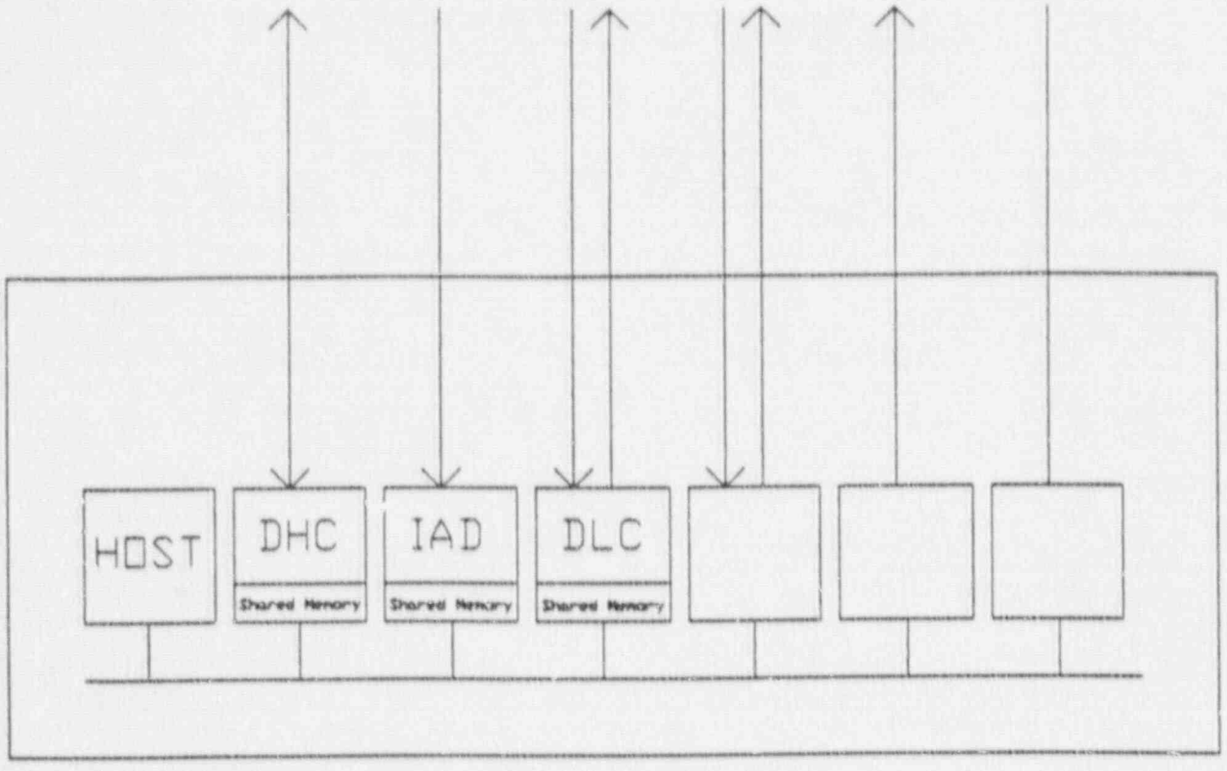


Figure 5-17: Integrated Protection Cabinet Front Layout

Functional Decomposition



- In each subsystem, the processing is distributed over multiple processors, typically one host processor and several slave processors.
- The most processing intensive I/O functions have been moved to slave processors which communicate with the host via shared memory.

Slave Processors

- Intelligent A/D (IAD): Digital filtering of analog inputs.
- Datalink Controller (DLC): Point-to-point simplex datalinks.
- Data Highway Controller (DHC): Multipoint data highway.

REACTOR COOLANT SYSTEM FLOW

LOOP 1

CH	TAG
1	BB-FT0416
2	BB-FT0417
3	BB-FT0418
4	BB-FT0419

LOOP 2

CH	TAG
1	BB-FT0426
2	BB-FT0427
3	BB-FT0428
4	BB-FT0429

LOOP 3

CH	TAG
1	BB-FT0436
2	BB-FT0437
3	BB-FT0438
4	BB-FT0439

LOOP 4

CH	TAG
1	BB-FT0446
2	BB-FT0447
3	BB-FT0448
4	BB-FT0449

FIELD

PS I/E

35/A TO ESF-1

35/B TO ESF-1

35/C TO ESF-1

35/D TO ESF-1

TI

IPC SIGNAL CONDITIONING

A/D

EU

TP 0501

LOW

TP 0502

QC

TP 0503

TO

54/A

REACTOR COOLANT FLOW LOOP 1 LOW PARTIAL REACTOR TRIP

TP 0504

LOW

TP 0505

QC

TP 0506

TO

54/B

REACTOR COOLANT FLOW LOOP 2 LOW PARTIAL REACTOR TRIP

TP 0507

LOW

TP 0508

QC

TP 0509

TO

55/A

REACTOR COOLANT FLOW LOOP 3 LOW PARTIAL REACTOR TRIP

TP 0510

LOW

TP 0511

QC

TP 0512

TO

55/B

REACTOR COOLANT FLOW LOOP 4 LOW PARTIAL REACTOR TRIP

REACTOR TRIP GROUP 1

DYNAMIC TRIP BUS

NOTES:

SIZEWELL B IPC
REACTOR COOLANT FLOW
SWB-PPS-0005
Rev: B 04SEP92 Sh: 09
Westinghouse Proprietary

OR

FUNCTION WHICH PRODUCES A TRUE OUTPUT WHEN ONE OR MORE INPUTS ARE TRUE

AND

FUNCTION WHICH PRODUCES A TRUE OUTPUT ONLY WHEN EVERY INPUT IS TRUE

NOT

FUNCTION WHICH PRODUCES A TRUE OUTPUT WHEN THE INPUT IS FALSE

BISTABLE

BISTABLE FOR WHICH THE OUTPUT IS TRUE WHEN THE INPUT VARIABLE IS GREATER THAN THE SETPOINT

BISTABLE

SAME AS ABOVE WITH HYSTERESIS

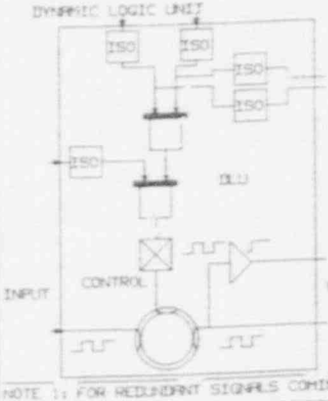
BISTABLE

BISTABLE FOR WHICH THE OUTPUT IS TRUE WHEN THE INPUT VARIABLE IS LESS THAN THE SETPOINT

BISTABLE

SAME AS ABOVE WITH HYSTERESIS

DYNAMIC LOGIC UNIT



THE INPUT PULSE TRAIN IS PASSED ON TO THE OUTPUT ONLY WHEN THE "CONTROL" SIGNAL IS TRUE

VARIABLE S TPOINT BISTABLE

BISTABLE FOR WHICH THE OUTPUT IS TRUE WHEN THE INPUT VARIABLE IS GREATER THAN THE VARIABLE SETPOINT

CONTACT INPUT

C1 SIGNAL CONDITIONING FUNCTION FOR INPUT OF CONTACT OPEN-CLOSED SIGNALS

DIGITAL INPUT

DI COMPUTER DIGITAL INPUT FUN'TION WHICH PERFORMS ERROR CHECKING FOR COMPLEMENTARY INPUT.

ISOLATED DIGITAL INPUT

IDI COMPUTER ISOLATED DIGITAL INPUT FUNCTION WHICH PERFORMS SIGNAL VALIDITY CHECKS THROUGH THE USE OF READBACK AND TOGGLE SIGNALS.

DIGITAL OUTPUT

DO COMPUTER DIGITAL OUTPUT FUNCTION.

TRIP OUTPUT

TO COMPUTER DIGITAL TRIP OUTPUT FUNCTION WITH READBACK. IF TRIP IS OUTPUT AND READBACK DOES NOT AGREE, DEADMAN IS NOT UPDATED. LOSS OF DEADMAN CAUSES FUNCTION TO ASSUME TRIP STATE.

QUALITY CHECK

QC C THE OUTPUT ASSUMES THE STATE SPECIFIED BY THE LOWER LETTER (C-CONFIGURABLE, T-TRUE, F-FALSE) IF BAD QUALITY IS PRESENT FOR ANY UPSTREAM FIELD SIGNALS. C SPECIFIES THAT THE STATE WILL ASSUME THE CONFIGURED STATE (TRUE, FALSE, PREVIOUS STATE, OR INPUT STATE).

INDICATOR LAMPS

(A) ACTUATION STATUS

(P) PERMISSIVE STATUS

(T) TRIP STATUS

(B) BYPASS STATUS

(V) VETO STATUS

TIME DELAY ON

TD ON OUTPUT IS CHANGED TO TRUE AFTER THE INPUT HAS BEEN TRUE FOR A DEFINITE INTENTIONAL TIME DELAY. OUTPUT CHANGES TO FALSE IMMEDIATELY UPON RECEIVING A FALSE INPUT.

TIME DELAY OFF

TD OFF OUTPUT IS CHANGED TO FALSE AFTER THE INPUT HAS BEEN FALSE FOR A DEFINITE INTENTIONAL TIME DELAY. OUTPUT CHANGES TO TRUE IMMEDIATELY UPON RECEIVING A TRUE INPUT.

2 OUT OF 4

2/4 A FUNCTION WHICH PRODUCES A TRUE OUTPUT WHEN 2 OR MORE INPUTS ARE TRUE

2 OUT OF 3

2/3 A FUNCTION WHICH PRODUCES A TRUE OUTPUT WHEN 2 OR MORE INPUTS ARE TRUE

3 OUT OF 4

3/4 A FUNCTION WHICH PRODUCES A TRUE OUTPUT WHEN 3 OR MORE INPUTS ARE TRUE

2 OUT OF 4 BYPASS

2/4 BYP — (TP)

A FUNCTION WHICH PRODUCES A TRUE OUTPUT WHEN 2 OR MORE INPUTS ARE TRUE. THIS FUNCTION ALSO RECEIVES BYPASS INPUTS WHICH MODIFY THE VOTING (SEE TABLE). THE FOLLOWING INFORMATION PRODUCED BY THIS FUNCTION MAY BE PROVIDED VIA THE TEST POINT INDICATED WITH A SUFFIX AS LISTED.

A. WHEN TWO OR MORE BYPASSES EXIST A MULTIPLE BYPASS ALARM IS PRODUCED.
B. TRIP ENABLE SIGNAL
C. REACTOR TRIP SIGNAL
D. GLOBAL TRIP SIGNAL

	NUMBER OF ASSOCIATED CHANNELS BYPASSED			
	1	2	3	4
VOTING	2/3	1/2	OUTPUT=TRUE	OUTPUT=TRUE

RETENTIVE MEMORY

S R A MEMORY FUNCTION WHICH RETAINS ITS OUTPUT STATE UPON THE OCCURRENCE OF AN EVENT CAUSING A RESTART (LOSS OF POWER, MANUAL RESTART, ETC.). THE MEMORY FUNCTION RETURNS FROM THE EVENT CAUSING A RESTART USING THE RETAINED PREVIOUS OUTPUT STATE IN THE DETERMINATION OF THE NEW OUTPUT STATE. THIS DETERMINATION AND ALL SUBSEQUENT DETERMINATIONS OF OUTPUT STATES ARE MADE ACCORDING TO THE TABLE SHOWN FOR THE OFF RETURN MEMORY COMMON FUNCTION.

OFF RETURN MEMORY

S R A MEMORY FUNCTION WHICH DOES NOT RETAIN ITS OUTPUT STATE UPON THE OCCURRENCE OF AN EVENT CAUSING A RESTART (LOSS OF POWER, MANUAL RESTART, ETC.). THE MEMORY FUNCTION RETURNS FROM THE EVENT CAUSING A RESTART IN THE OFF STATE OR FALSE. DURING OPERATION THE OUTPUT STATE IS DETERMINED ACCORDING TO THE TABLE BELOW:

S (SET)	R (RESET)	PREVIOUS OUTPUT STATE	OUTPUT STATE
FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	DO NOT CARE	FALSE
TRUE	FALSE	DO NOT CARE	TRUE
TRUE	TRUE	DO NOT CARE	FALSE

NOTE 1: FOR REDUNDANT SIGNALS COMING FROM VARIOUS SUBSYSTEMS, THE COMMUNICATION SUBSYSTEM SHALL SELECT THE SIGNAL WITH THE HIGHEST PRIORITY AND GOOD QUALITY TO BE SENT TO THE EXTERNAL SYSTEMS. THE ORDER OF PRIORITY IS COMM, RT1, RT2, ESF1, AND ESF2. ADDITIONAL INFORMATION IS SENT ALONG WITH THE SELECTED SIGNAL TO INDICATE THE QUALITY AND THE SUBSYSTEM ORIGIN OF THE SELECTED SIGNAL, WHETHER ANY OF THE REDUNDANT SIGNALS HAD BAD QUALITY, AND WHETHER ANY COMPARISON FAILURES EXIST BETWEEN REDUNDANT SIGNALS WITH GOOD QUALITY.

SIZEWELL B IPC
SYMBOLS
SWB-PPS-0005
Rev: B 04SEP90 | SH: 03
Westinghouse Proprietary

R TO E

R/E — SIGNAL CONDITIONING FUNCTION WHICH CONVERTS A RESISTANCE TO A VOLTAGE SIGNAL. INCLUDES SURGE FILTERING.

I TO E

PS I/E — SIGNAL CONDITIONING FUNCTION WHICH CONVERTS A 4-20 mA SIGNAL TO A VOLTAGE SIGNAL. INCLUDES SURGE FILTERING. A POWER SUPPLY IS ALSO PROVIDED

I TO E

I/E — SIGNAL CONDITIONING FUNCTION WHICH CONVERTS A 4-20 mA SIGNAL TO A VOLTAGE SIGNAL. INCLUDES SURGE FILTERING

ANALOG TO DIGITAL

A/D — COMPUTER ANALOG INPUT AND ANALOG TO DIGITAL CONVERSION FUNCTION. EXCEPT FOR NI, ALSO INCLUDES DIGITAL FILTERING.

DIGITAL TO ANALOG

D/A — COMPUTER ANALOG OUTPUT AND DIGITAL TO ANALOG CONVERSION FUNCTION INCLUDES CONVERSION FROM ENGINEERING UNITS TO DIGITAL VALUE

ELECTRICAL AND ENGINEERING UNIT CONVERSION

EU — CONVERSION FROM A/D UNITS TO ELECTRICAL AND ENGINEERING UNITS, AND PERFORMS STATUS CHECKING.

LAG

L — LAG FUNCTION

LEAD/LAG

L/L — LEAD/LAG FUNCTION

RATE/LAG

R/L — RATE/LAG FUNCTION

SENSOR

FIELD INSTRUMENTATION

HARDWIRED INFORMATION TRANSMISSION

ANALOG

LOGIC

SOFTWARE INFORMATION TRANSMISSION

NUMERIC

LOGIC

MULTIPLE (NUMERIC AND/OR LOGIC)

MULTIPLEXED DATA

CONNECTION BETWEEN PAGES. THE NUMBER INDICATES THE PAGE AND THE LETTER INDICATES THE CONNECTION POINT ON THE PAGE.

Tx — FUNCTION WHICH TRANSMITS INFORMATION MULTIPLEXED

Rx — FUNCTION WHICH RECEIVES INFORMATION MULTIPLEXED AND PERFORMS ERROR CHECKING

BYP — MANUAL BYPASS

TRIP — MANUAL TRIP

E TO I

E/I — SIGNAL CONDITIONING FUNCTION WHICH CONVERTS AN ELECTRICAL SIGNAL TO A 4-20 mA SIGNAL

E TO O

E/O — SIGNAL CONDITIONING FUNCTION WHICH CONVERTS AN ELECTRICAL SIGNAL TO AN OPTICAL SIGNAL

O TO E

O/E — SIGNAL CONDITIONING FUNCTION WHICH CONVERTS AN OPTICAL SIGNAL TO AN ELECTRICAL SIGNAL

TEST POINT

TP — MONITORING POINT DISTRIBUTED TO AUTO TESTER OR COMMUNICATION

TEST INJECTION

TI — TEST INJECTION POINT FOR AUTO TESTER

SUM

MULTIPLEXER

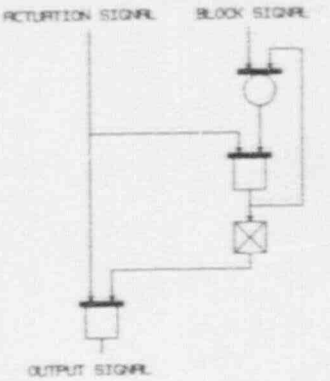
MUX

LAMP DRIVER

LD — OUTPUT FOR DRIVING LAMPS

ACTUATION BLOCK

A B — A FUNCTION WHICH ALLOWS BLOCKING OF THE ACTUATION SIGNAL, AS INDICATED BELOW



SQUARE ROOT

ISOL — ISOLATING FUNCTION

CONTACT

SW — FIELD CONTACT, SEPARATION GROUPS REQUIRE PHYSICALLY SEPARATE CONTACTS

REFERENCES:

1. PPS LOGIC DIAGRAMS SKB-SB-JI4032 REV 3 29-MAY-1990

SIZE/CELL B IPC SYMBOLS SKB-PPS-0005	
Rev: B 24SEP90	Sht: 02
Westinghouse Proprietary	

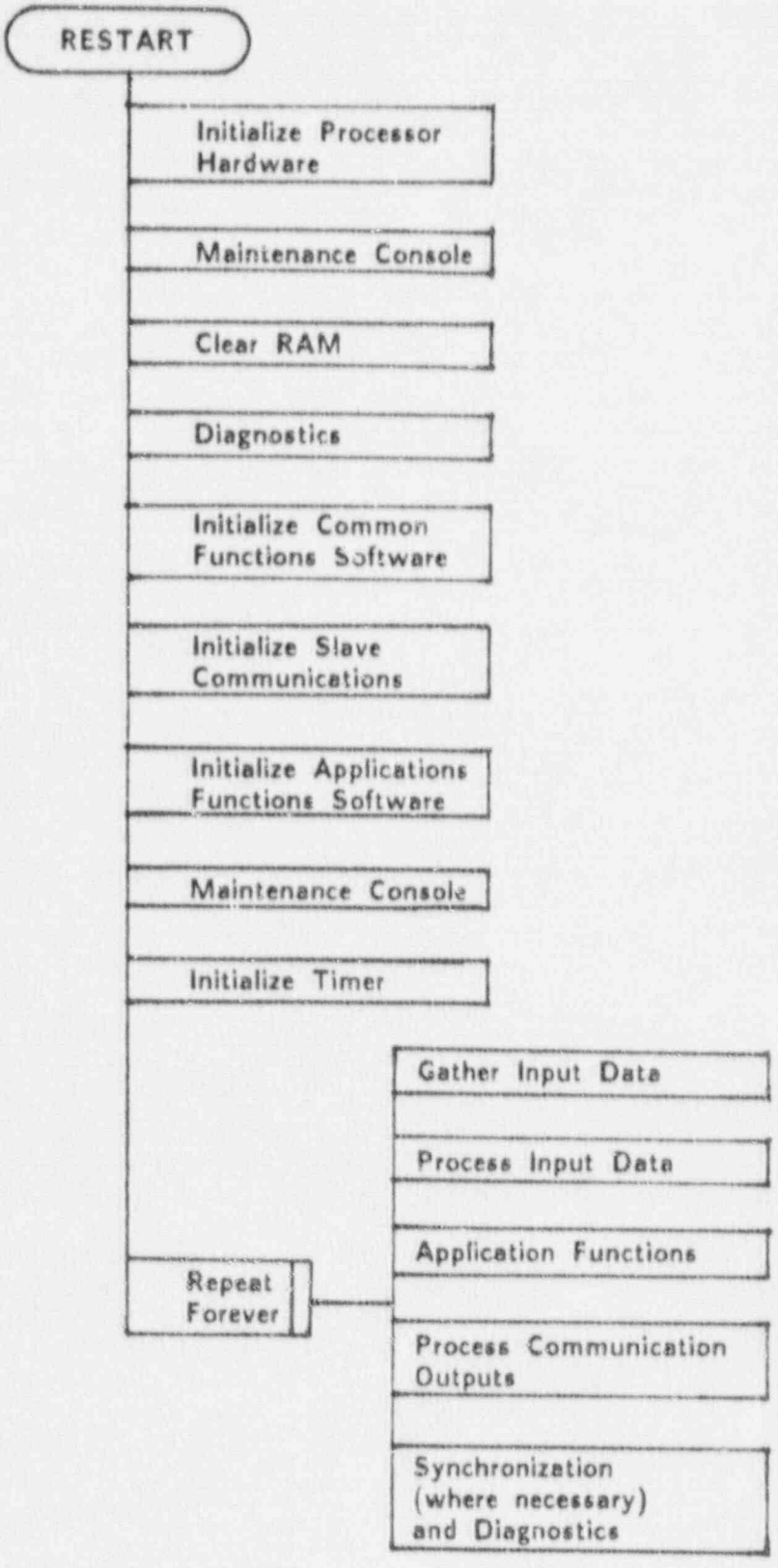
Software Design Constraints

"Certain general constraints are imposed on the characteristics of that software which provides an essential protection or control function."

These constraints are consistent with the 414 IPS design philosophy. The standards represent a formalization of this philosophy, along with additional thinking based on new conditions and capabilities.

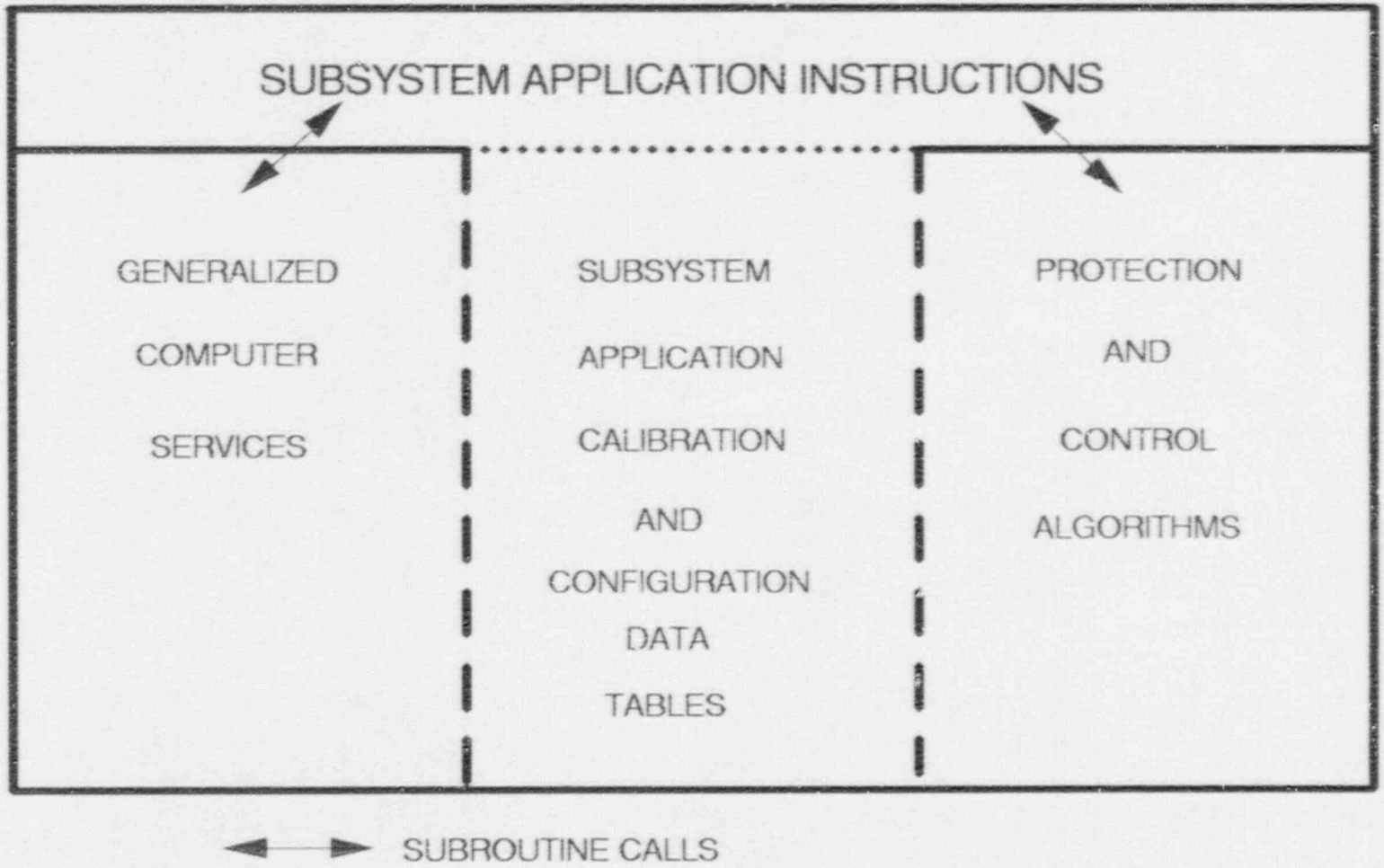
SOFTWARE DESIGN CONSTRAINTS

- INTERRUPTS
- CONCURRENT ACCESS
- MULTIPLE PROCESSORS
- RE-ENTRANCY
- MODULARITY
- PROCEDURE STRUCTURE
- DATA BOUNDING
- APPLICATION VS. SYSTEM
- CODE VS. DATA
- HIGH LEVEL LANGUAGE

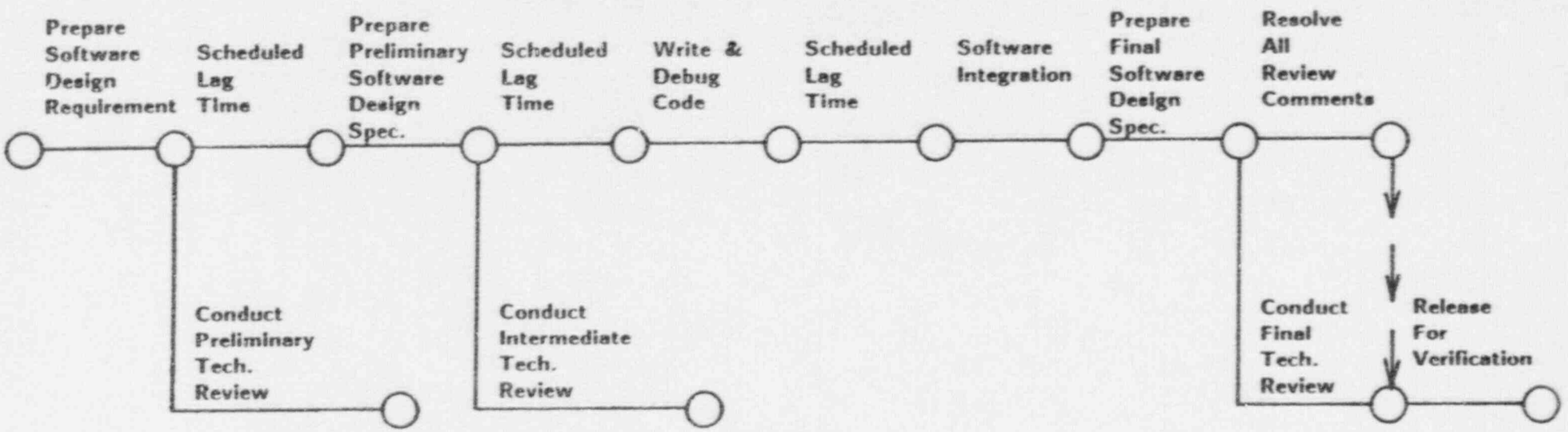


Top-Level Software Structure

IPS AND ICS APPLICATIONS SPECIFIC AND GENERALIZED SOFTWARE MODULES



Software Module Development



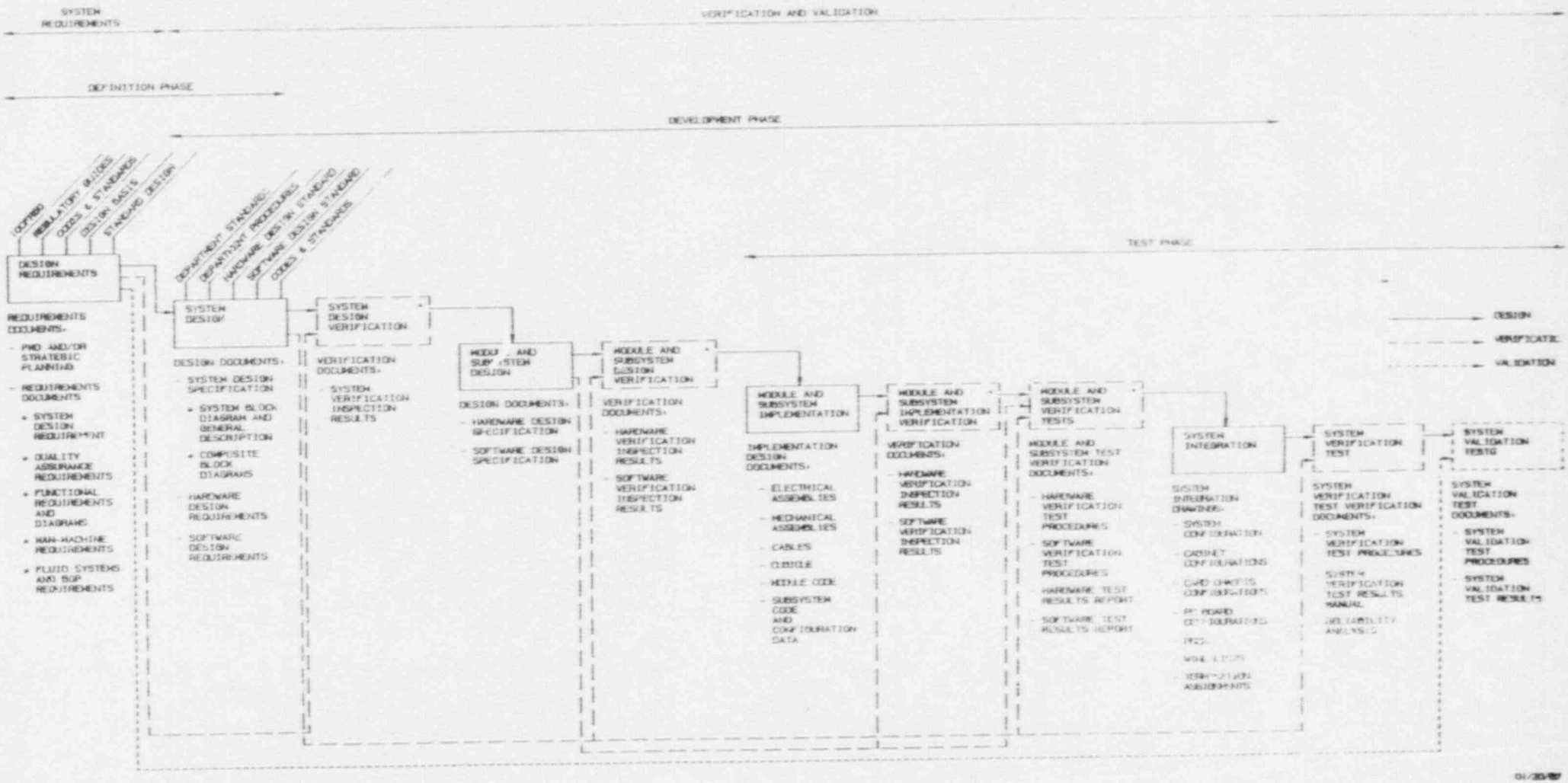
Westinghouse Electric Corporation

SOFTWARE V&V REQUIREMENTS

J. B. REID

File:ACRS10.WK1

SYSTEM DEVELOPMENT/IMPLEMENTATION PROCESS (SYSDIP)



• VERIFICATION REQUIRED FOR SAFETY SYSTEMS ONLY

V&V GUIDELINES, CODES, & STANDARDS

ANSI/IEEE-ANS-7.4.3.2-1982

Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Plants

IEC PUBLICATION 880

Software for Computer Systems in the Safety Systems of Nuclear Power Stations

IEEE STANDARD 603

Standard Criteria for Safety Systems of Nuclear Power Generating Stations

IEEE STANDARD 730

Standard for Software Quality Assurance Plans

IEEE STANDARD 829

Standard for Software Test Documentation

IEEE STANDARD 1012

Standard for Software Verification and Validation Plans

VERIFICATION AND VA .IDATION PHILOSOPHY

BOTTOM UP VERIFICATION TESTING APPROACH

Hardware and Software modules are individually tested in depth

STRESSING THE DESIGN

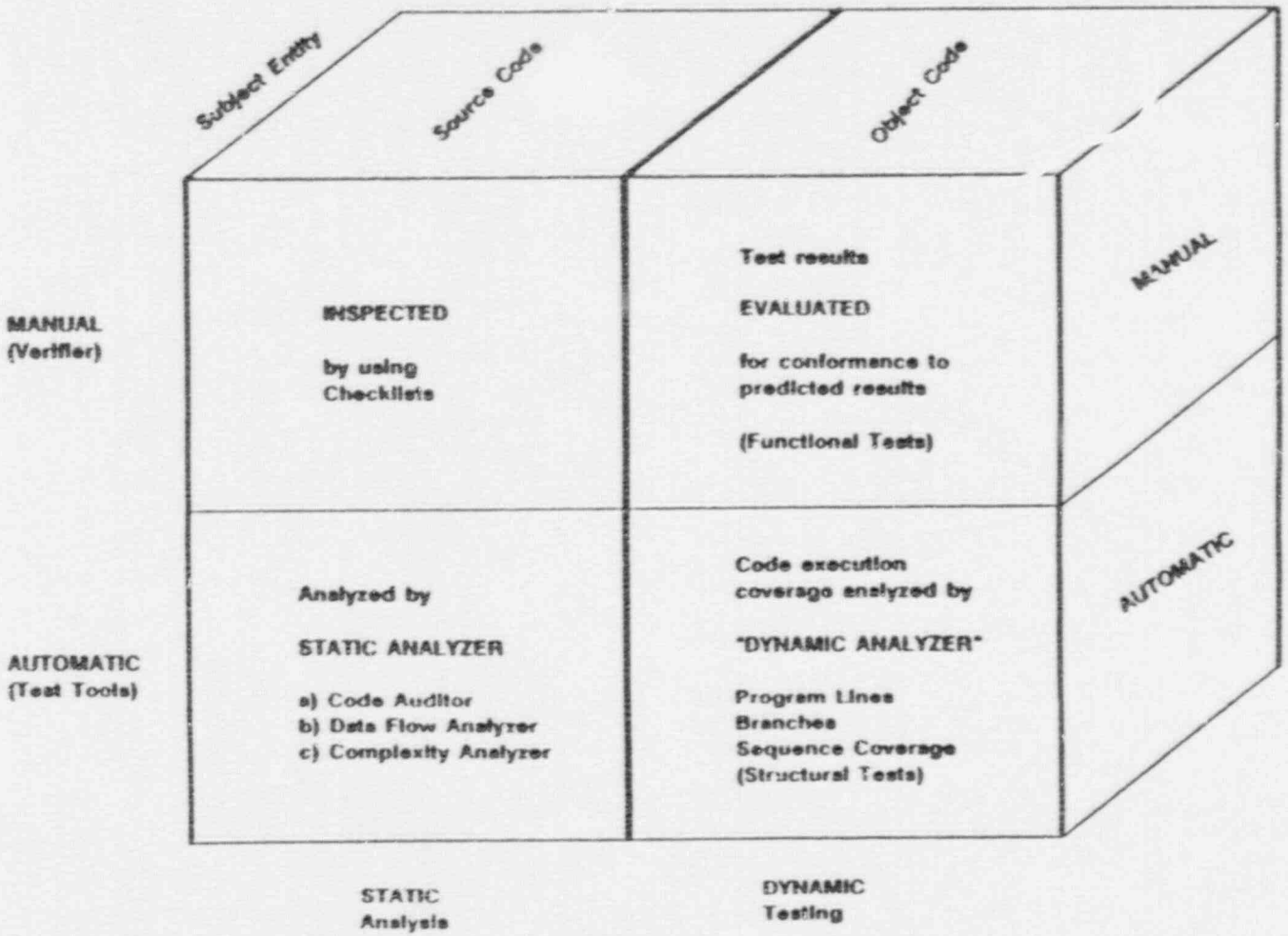
By testing each module over its possible range of use, a higher level of assurance is achieved over any testing that could be done at the integrated system level.

ANOMALY REPORTING

Anomaly reports provide an auditable demonstration of the completeness of the verification, and the disposition of the issues raised by the verifiers.

IPS – Design Verification & Validation

Software Verification Model



86702-1-10

SOFTWARE SECURITY REQUIREMENTS

SOFTWARE SECURITY REQUIREMENTS

- PERIODIC TESTING
- BUILT-IN DIAGNOSTICS
- EMBEDDED CHECKSUMS
- READ ONLY MEMORIES
- DOOR LOCKS
- LIMITED PHYSICAL ACCESS
- LIMITED SOFTWARE ACCESS

DIAGNOSTIC SOFTWARE AND HARDWARE FOR CRITICAL REAL-TIME SYSTEMS

M. D. Bowers, J. P. Arnold, and A. W. Crew

Engineering Technology Division
Westinghouse Electric Corporation
Research & Development Center
1310 Beulah Road
Pittsburgh, PA 15235
(412) 256-2456/2601/2539

R. J. Gibson and W. D. Ghrist III

Nuclear and Advanced Technology Division
Westinghouse Electric Corporation
P.O. Box 598
Pittsburgh, PA 15230
(412) 733-6540/6343

Abstract

Many techniques have been developed to deal with the various issues inherent in the fault-tolerant design of critical real-time systems. Central to these techniques is a defense-in-depth philosophy, in which different layers of the design address both different and overlapping fault detection and recovery issues. The addition of microprocessor-based technology offers a new opportunity to extend the defense-in-depth philosophy for critical real-time systems, particularly in the nuclear industry. Traditionally, in protection systems for commercial nuclear applications, a complete off-line functional test of the system was performed. Now, embedded self-diagnostics can provide a continuous test of the system to speed fault identification and repair. These embedded diagnostics are an additional layer of defense which was never before possible. Since the embedded self-diagnostics run while the system is performing the critical function, it is desirable to keep them as simple as possible. An attempt to detect every possible fault would increase total system complexity and decrease response time. Thus, the embedded diagnostics have been designed to detect the more probable faults quickly. Safety is never compromised since complete functional tests are still performed on a periodic basis.

This paper describes a library of diagnostic algorithms. In addition, a dedicated diagnostics board, the Multibus Diagnostic Monitor, will be described.

Introduction

This paper describes a diagnostic software library which contains algorithms to test read-only memory, read/write memory, address lines, the main processor instruction set, the numeric data processor instruction set, and mutual exclusion hardware. The software library also contains algorithms to respond to unexpected software interrupts. In addition to the diagnostic software library, a dedicated subsystem diagnostics board called the Multibus Diagnostic Monitor (MDM) will also be described.

Diagnostic Software

A library of diagnostic algorithms for use in real-time microprocessor-based systems has been developed. The major design goals are to detect the most common failures as quickly as possible and to detect a majority of the less common failures in a timely manner. Note: Some highly unlikely failures may still go undetected by these algorithms, but the algorithms are only the lowest layer of protection. All of our critical systems have fault tolerance (usually redundancy) at higher levels in the system so that even these unlikely failures can be detected and will not compromise safety.

The algorithms are intended to be used with Intel Corporation's 8086 family of microprocessors. This includes the 8086, 8088, 80186, 80188, and 80286 (in real mode only). The algorithms are *not* compatible with 80286 protected-mode operation.

The diagnostic algorithms were developed to detect the failures in several hardware devices. The following is a list of these devices; general descriptions of the algorithms used to detect corresponding failures are given on the following pages:

- Read-only memories containing software, configuration or calibration data.
- Read/write memories containing program variables and data.
- Address lines addressing read/write memories.
- Main processors.
- Numeric Data Processors.
- Hardware used to mutually exclude multiple processors from the same shared-memory resource.
- Interrupt hardware.

The uniqueness of the algorithms used to test these devices lies in their structure. The structure was dictated by the system

time constraints in which the software is to run. For these systems, start-up time is relatively unrestricted, while run-time processing consists of time-restricted cycles. The cycles usually contain a large amount of application specific processing, with the remaining time being used for diagnostic testing. For this reason, the algorithms were developed such that all of a particular resource could be tested at once (such as during system start-up) or incrementally (during run-time cycles). In the case of the memory and address line diagnostic algorithms, configuration tables are used to define the locations of the regions of memory, or combinations of address lines to be tested. Each incremental algorithm tests a subset of the regions or combinations. Similarly for the instruction set diagnostic tests, the incremental algorithms test a subset of the instruction sets. In all cases, the size of the subset to test is an argument to the incremental algorithm; this value could be determined by the amount of time left in a current run-time cycle. The complete start-up diagnostic tests are implemented by calling the incremental algorithms repetitively until all of the corresponding resource is tested.

Unlike the incremental algorithms, the algorithms used to detect failures of mutual exclusion hardware and interrupt hardware do not test "blocks" (regions, combinations, or sets) of their corresponding resources. Instead, the mutual exclusion algorithm is designed to be implemented once a cycle for every memory resource shared by a processor. For the case of unexpected software interrupts, interrupt handlers are installed during system start-up; run-time diagnostic processing is not necessary.

For all the diagnostic algorithms, failures are reported in the same way: Upon the detection of a failure, diagnostic information describing the kind of failure and the location of the failure is immediately copied to dedicated memory locations. These locations usually exist in the shared memory of another processor; this second non-failing processor has the responsibility of propagating the failure report outside the subsystem or saving the information in non-volatile memory for examination at a later time. In the case where the information is reported to a non-failing processor which propagates the failure report outside the subsystem, the identification of the subsystem and the failed processor is included in the report. After failure information has been reported by the failing processor, it enters its shut-down or halted state. Non-maskable interrupts can bring the processor out of this state (which is not desirable); consequently, the non-maskable interrupt must be handled by a failure reporting routine that subsequently re-enters the halted state.

Read-only Memory Diagnostic Algorithms

Read-only memory devices are tested by summing all the memory words (a word is two bytes or 16 bits) in a particular region; when all the words have been summed, the final sum is verified against an expected checksum previously-computed by an independent system. Configuration of this algorithm is achieved through a table that describes the regions to be tested and the locations of the previously-computed checksums. The incremental algorithm sums and verifies against the expected checksum when appropriate.

More exactly, the incremental algorithm sums the number of words requested by its argument and saves the intermediate sum until, after successive increments are summed, the end of a region is reached. When the end of a region is reached, the sum is verified at that point, and the intermediate sum is then cleared. The next words to sum are words in the next region specified by configuration. Figure 1 presents an example for the incremental algorithm. In this figure, Num_Words_to_Sum represents the argument to the algorithm - the size of the subset to test.

```
REPEAT the following loop until Num_Words_to_Sum is zero.
  IF Num_Words_to_Sum is less than the number of words left in
  the current region, THEN
      SUM the number of words specified by Num_Words_to_Sum
      SAVE this intermediate sum for next time.
      SET Num_Words_to_Sum equal to zero.
  ELSE
      SUM the number of words left in the current region.
      VERIFY this sum against its corresponding expected
          checksum.
      IF the sum did not verify, THEN
          immediately REPORT the failure, and HALT the
              processor.
      END IF
      SUBTRACT the number of words left in the region
          from Num_Words_to_Sum.
      MAKE the next region the current region.
  END IF
  CONTINUE the REPEAT loop IF Num_Words_to_Sum is more
      than zero.
END of the REPEAT loop.
```

Figure 1: Example Incremental Memory Checksum Algorithm

Read/Write Memory Diagnostic Algorithms

There are two basic types of read/write memory failures: 1) The failure of a memory cell, data line, or address line that is stuck in a particular state (always 1, or always 0); or 2) The failure of a memory cell, data line, or address line that is in a particular state because it is "coupled" to one or more other memory cells or data lines. Coupling faults usually occur between physically-adjacent cells or lines.

While a test for stuck memory cells or data lines needs only to write and read both a state and its complement to that cell or line, detecting coupling faults can only be detected by writing all combinations of bit patterns to all combinations of groups of memory cells or lines and then checking all the other cells or lines for interference. One way to reduce the complexity of full coupling fault tests would be to use groups of memory cells or data lines that are physically adjacent to a cell or data line to be tested.

However, RAM chips can have physically different architectures yet have identical pin outs. In time, changes in technology mean even more different physical architectures for replacement chips. Therefore, no consistent physical architecture can be assumed throughout a typical RAM-based system. Physically adjacent or "neighboring" cells cannot be easily identified and no assumptions can be made that might reduce the number of coupling faults to faults that only occur between neighboring cells. Therefore, tests for coupling faults in memory cell arrays must check for interference between a cell and all other cells. This kind of diagnostic function is intended for chip fabrication time diagnostics and is not suitable or practical for run-time system diagnostics.

On the other hand, data line and address line coupling faults can be detected much more practically: The number of data and address lines is small compared to the number of memory cells and the number of possible interference patterns is reduced.

The read/write memory diagnostic algorithms described in

this section detect memory cells (bits) and memory data lines that are stuck in a particular state. They also detect coupling failures between memory data lines (coupling failures occur when the state of one or more memory cells or data lines affect other cells or lines). Although the read/write diagnostic algorithms modify memory locations, they are considered non-destructive tests because the algorithm returns the locations to their original values upon completion of the test. Configuration of these algorithms is achieved through use of a table that describes the regions of memory to be tested. The algorithm used to incrementally test read/write memory tests a subset of one or more of these regions.

A data line coupling fault test needs only to compare written and read values (data line patterns) at any address. This test has been combined with the test for stuck memory cells by using the data line patterns (and their complements) as bit patterns for detecting the stuck cells. (The test would therefore test all the bits in a memory word instead of only testing one bit as was needed to accomplish a test of a memory cell.)

At the heart of the read/write diagnostic algorithm is a low-level algorithm used for testing two adjacent words or two adjacent bytes at a time. For the purposes of this paper, an algorithm that tests two words at a time is described. An algorithm for testing two adjacent bytes can be derived from a simple extension of the algorithm described. The use of either one of these algorithms depends upon the width (word-wide or byte-wide) of the data bus and the memory devices.

If interrupts are enabled, they are locked out for critical portions of the test. The test can not be run on RAM which is subject to DMA, dual-port access, or access in response to non-maskable interrupts.

Provisions are made after each memory write to prevent inaccurate test results due to storage of the test patterns on the stray capacitance of data lines. Therefore, after every write of a test pattern and before the verifying read, the complement of the pattern is written to or read from a different location. This resets the data lines so that the verifying read will not provide misleading results. The most convenient way to satisfy this requirement is to use the next word as the different location. By writing a test pattern to the first of two words to test, writing the complement of the pattern to the second word, reading the first word to verify, and finally reading the second word to verify, these requirements are satisfied. At the same time, the two adjacent words have been tested.

In order to test for possible coupling faults, different test patterns must be used on the two memory locations. A series of test patterns that contains all 64K (2^{16} for 16 data lines) different word values must be used to detect all 64K possible coupling faults. However, the test patterns that detect more probable coupling faults should be used more often than test patterns that detect less probable coupling faults. The most probable coupling faults occur between physically-adjacent data lines. These lines are assumed to be logically adjacent. The most probable coupling faults are detected by using a word test pattern consisting of alternating bits (for example, "5555" hexadecimal).

Thus, the series of test patterns used by the read/write diagnostic algorithm is such that as the memory regions specified in the configuration tables are tested over and over again, and the series of patterns is repeated over and over again, every word in all the regions is tested with every word test pattern in the series. To accomplish this, the total number of words to test in all the regions should not be a multiple of the number of test patterns in the series. For this purpose, the series of test patterns consists of all 64K possible patterns with the "5555" hexadecimal pattern used every other time - with the exception of when the series repeats. After the "FFFF" hexadecimal pattern is used, a test pattern of "0000" hexadecimal is used without using "5555" in between. The test series consists of: "0000", "5555", "0001", "5555", "0002", "5555",... "5555", "FFFF", "0000", "5555", "0001", "5555",... "5555", and "FFFF". This makes the total length of the series of test patterns equal to $2*2^{16} - 1$, or 131,071 (a prime number). Each test pattern is used to test two words. Since the length of the series is a prime number, the probability that the total size of memory regions to be tested being a multiple of the length of the test pattern series is small.

Before any word memory locations are tested, the previous values of those locations are saved in microprocessor registers and then restored after the test of those locations is complete. The restoration is attempted even in the event that a failure is detected - in case those locations held vital information needed to report the failure.

Figure 2 presents a sample algorithm that satisfies the requirements described above. In this example, a local variable, Test_Pattern, is used to specify the word value to use to test the words. The sample algorithm assumes that the number of words tested is even. Also, the sample algorithm uses the following intermediate variables to maintain which test pattern is to be used next.

Pattern_Counter: A word value that is incremented with each use and generates all possible word test patterns. (Assume, for the example, that this variable has been initialized to zero.)

Used_Alternating_Bits_Last: A Boolean flag that indicates "true" if the alternating bit pattern was used as the last test pattern. The flag indicates "false" if the last test pattern was generated from the Pattern_Counter. (Assume, for the example, that this variable has been initialized to false.)

Address Line Diagnostic Algorithms

The address line diagnostic algorithms detect both failures of address lines that are stuck in a particular state and coupling faults between address lines. Coupling faults between address lines are assumed to be modeled by shorts between physical address lines. As described below, configuration for these algorithms describes the combinations of address lines that can be tested.

In order to detect a failure in an address line, two addresses must first be generated that model the failure: A "failure model" address where the suspect address line is in the state that would model the failure, and a "base line" address where all the lines are the same as in the "failure model" address - except for the suspect address line; the suspect address line in the "base line" address must be in the correct state - the inverse of the modeled failure state. For example, if the function was to detect a "stuck at one" failure of the fifth address line, the binary representation of the two addresses would be

Failure Model:

(MSB) XXXX XXXX XXXX XXX1 XXXX (LSB)

```
IF Used_Alternating_Bits_Last
OR IF Pattern_Counter is zero, THEN
    SET Test_Pattern equal to Pattern_Counter
    SET Used_Alternating_Bits_Last to false.
    INCREMENT Pattern_Counter by one (for its next use).
ELSE
    SET Test_Pattern equal to "5555" hexadecimal.
    SET Used_Alternating_Bits_Last to true.
END IF
COMPUTE the addresses of the next two words to test.
Temporarily SAVE the previous values of the two words to test.
WRITE Test_Pattern at the address of the first of the two words
    to test.
WRITE the complement of Test_Pattern at the address of the second
    word. (The second write clears the data lines after the
    first write.)
READ the values in the order written.
    (The first read clears the data lines after the second write.)
VERIFY that the respective patterns were stored correctly.
IF a failure occurred, THEN
    Immediately RESTORE the previous values of the tested words.
    REPORT the failure, and HALT the processor.
END IF
WRITE the complement of Test_Pattern at the address of the first
    word.
WRITE Test_Pattern at the address of the second word.
    (Again, the second write clears the data lines after the
    first write.)
READ the values in the order written.
    (The first read clears the data lines after the second write.)
VERIFY that the respective patterns were stored correctly.
IF a failure occurred, THEN
    Immediately RESTORE the previous values of the tested words.
    REPORT the failure, and HALT the processor.
END IF
RESTORE the previous values of the tested words.
```

Figure 2: Example Lower Level Read/Write Memory Test

Base Line:

(MSB) XXXX XXXX XXXX XXX0 XXXX (LSB)

where "X" represents the states of the other address lines. The states of the other lines are inconsequential to the test for detecting address lines that are stuck, but as described below, they can be used to also detect lines that are shorted together.

In order to detect an address line that is in a particular failure state because it is shorted to one of the other address lines that can be tested in an address region, the "failure model" address must consist of all the address lines that can be tested set to the same state – the particular failure state. As before, all the address lines in the "base line" address must be the same as in the "failure model" address except for the suspect address line; the suspect address line in the "base line" address must be in the correct state – the inverse of the particular failure state. As a second example, if the lower 14 lines can be tested and if the function was to detect whether the fifth address line was in its one state because it is shorted to the one of the other 14 address lines, the binary representation of the two addresses would be:

Failure Model:

(MSB) XXXX XX11 1111 1111 1111 (LSB)

Base Line:

(MSB) XXXX XX11 1111 1110 1111 (LSB)

where "X" represents the states of the address lines that cannot be tested. The untested lines select the address line region itself as opposed to any other address line region of the same size. Notice that these two generated addresses detect whether the fifth address line is "stuck at one" or shorted to one of the other testable address lines.

Similarly, if the function were to detect whether the fifth address line was "stuck at zero", or shorted to another testable address line, the binary representation of the two generated addresses would be:

Failure Model:

(MSB) XXXX XX00 0000 0000 0000 (LSB)

Base Line:

(MSB) XXXX XX00 0000 0001 0000 (LSB)

As a conclusion to this discussion, in order to test each address line that can be tested, two sets of "failure model" and "base line" addresses are generated for each line. One set that is used to detect whether the line is "stuck at one" or shorted to another testable address line, and another set that is used to detect whether the line is "stuck at zero" or shorted to another testable address line.

After the "failure model" and "base line" addresses have been generated, the failure is detected by first writing a pattern at the "base line" address and then checking to see if that pattern was written at the "failure model" address instead. In order to detect whether the test pattern was written at the "failure model" address, the previous value at the "failure model" address must be different than the pattern written at the "base line" address. This is accomplished by first writing the complement of the test pattern to the "failure model" address.

Before any test is performed, the previous values of both locations are saved in microprocessor registers and then restored after the test is complete. The restoration is attempted even in the event that a failure is detected – in case those locations held vital information needed to report the failure.

Figure 3 presents an example algorithm that might be used to satisfy the requirements of the address line diagnostic algorithm. In this example, a local variable Line_To_Test is used to specify the number of the address line to test.

Configuration for these algorithms is achieved through the use of a table that describes the allowed combinations of address lines to be tested. In order to ensure that address lines are tested along their entire physical path, address line combinations that address each physical read/write memory device should be configured.

In order to describe the set of allowed combinations of address lines that can be tested, the term "address line region" is introduced. An address line region is a region of memory that is addressed by all combinations of a particular set of address lines. The smallest address line region would be a region of memory locations that are addressed by all combinations of one address line – the least significant address line. This address line region would consist of two contiguous bytes.

```
GENERATE the 'failure model' and 'base line' addresses
    needed to detect if Line_to_Test is stuck at one or shorted
    to one.
GENERATE the 'failure model' and 'base line' addresses
    needed to detect if Line_to_Test is stuck at zero or shorted
    to zero.
WITH each set of 'failure model' and 'base line' addresses
DO the following loop
    Temporarily SAVE the previous values at the two addresses.
    WRITE '55' hexadecimal at the 'failure model' address.
    WRITE 'AA hexadecimal at the 'base line' address.
    READ the value at the 'failure model' address and
    CHECK to ensure that '55' hexadecimal is still there.
    IF a failure occurred, THEN
        Immediately RESTORE the previous values at the
        tested addresses, REPORT the failure, and HALT
        the processor.
    END IF
    RESTORE the previous values at the tested addresses.
END of the DO loop.
```

Figure 3: Example Address Line Test Algorithm

The Intel 8086 family of microprocessors (including the 80286 operating in real mode) can address, at most, 1M byte of memory. To address all possible locations, 20 address lines are used. Thus, the largest address line region is the total address space of the processor. If n is the number of address lines to be tested (where n is an integer greater than zero and no greater than twenty), then the address line region consists of all the memory locations addressed by line 0 through line $n - 1$. Lines n and above select the address line region itself as opposed to other address line regions of the same size. For example: Memory addresses 20000H through 200FFH comprise an address line region of eight address lines. Memory addresses 20100H through 201FFH comprise another, separate address line region of eight address lines. Memory addresses 20000H through 207FFH comprise an address line region of 11 address lines. This 11-line region happens to contain both of the eight-line regions as subsets. Memory addresses 20800H through 20FFFH comprise a different 11-line region, one which does not contain the previously-described eight-line regions.

The configuration table for the address line diagnostic algorithm specifies a beginning address of an address line region, and the number of lines that can be tested for that region (n). For each physical bank of RAM, an address line region corresponding to that bank should be specified, if possible, in the address line configuration table.[1] This allows the address lines to be tested along their entire routes. For example, suppose the total RAM space of a processor was 64K bytes, and suppose the 64K bytes were divided into two 32K-byte physical banks of RAM. This is illustrated in Figure 4. The memory locations in these banks are selected by address lines 0 through 14 ($2^{15} = 32{,}768 = 32$ K). In order to test these address lines in each of the 32K-byte banks of RAM, an address line region corresponding to each bank would have to be specified in the configuration table. Address line 15 selects one of the banks as opposed to the other. In order to test line 15, an address line region corresponding to the whole 64K-byte bank of RAM must also be specified in the address line configuration table. Thus, the configuration table would contain three entries. If the 32K-byte banks were physically separated further into smaller banks, the configuration table would contain entries that corresponded to those smaller banks.

If only the larger 64K-byte bank of RAM were specified in the configuration table, then tests of combinations of address lines forming low addresses (the 15th address line would be clear) would test the lines along their routes into Bank 0. Tests of combinations of lines forming high addresses (the 15th address line would set) would test the lines along their routes into Bank 1. Therefore, all combinations of lines 0 through 14 with the 15th line being clear would not be tested along their routes into Bank 1. Similarly, all combinations of lines 0 through 14 with the 15th line being set would not be tested along their routes into Bank 0.

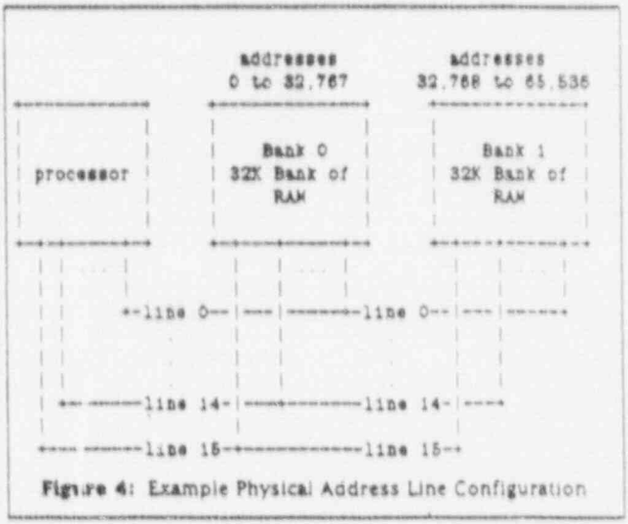


Figure 4: Example Physical Address Line Configuration

Main Processor Instruction Set Diagnostic Algorithms

The main processor instruction set diagnostic algorithms detect failures in the 8086 and 8088 class of microprocessors, in addition to the 80188, 80186, and 80286 (in real mode only) processors. Instruction set failures are detected by checking for correct results after executing a general set of microprocessor commands. The general set of commands will be divided into subsets of commands. The incremental main processor diagnostic algorithm tests one or more of these subsets, depending on the argument to the algorithm.

Since microprocessor address failures are usually independent of data failures, the testing of every command with every address mode is not necessary; addressing mode failures are detected by testing the addressing modes with a subset of the general set of commands. The general set of microprocessor commands, registers, and addressing modes tested are as follows. The commands are listed merely in similar groups, the groups are not required subsets.

- microprocessor registers,
- shifts, rotates, and logical operators,
- conditional jumps,
- stack operations,
- signed and unsigned byte-wide integer multiplications,
- signed and unsigned word-wide integer multiplications,
- signed and unsigned word-by-byte and double word-by-word divisions,

[1] Some of the regions that correspond to physical banks of RAM cannot be tested in their entirety because of shared-memory or DMA communication restrictions.

decimal and ASCII adjusts,

- conversion of bytes to words and words to double words,
- repeat operations and the byte and word string scans, load, stores, compares, and moves,
- the no operation command,
- indirect addressing with no displacement, and with 8- and 16-bit displacement, and
- segment override addressing.

The microprocessor commands that are *not* tested by this library of diagnostic routines are listed below. If these commands are implemented, they must be tested separately.

- software generated interrupts,
- translation, escape, and halt commands.

Numeric Data Processor Instruction Set Diagnostic Algorithms

The numeric data processor (NDP) diagnostics detect failures in the 8087 class of numeric data processors. Numeric data processor failures are detected by checking for correct results after executing a general set of numeric data co-processor functions. The general set of functions are divided into subsets of functions. The incremental numeric data co-processor diagnostic algorithm tests one or more of these subsets, depending on the argument to the algorithm.

The general set of numeric data processor functions that are tested is as follows. The functions are listed merely in similar groups, the groups are not required subsets.

- real addition and subtraction,
- real multiplication and division,
- the square root utility function,
- conversion of real numbers to integers and integers to real numbers,
- real comparisons,
- the masked response to an exception not configured to interrupt the CPU.

The general set of functions tested in the numeric data co-processor instruction set diagnostic algorithms is general for the 8087 class of co-processors. If commands outside this set are used, those commands and processors must be tested separately by the application that uses them.

Mutual Exclusion Hardware Diagnostic Algorithms

In multi-processor environments, information is typically exchanged between processors through regions of shared RAM memory. (Two or more processors may have read or write access to the same memory location.) It is obvious that one or more processors should be prevented from reading a block of memory locations that is currently being modified by another processor.

A system bus lock is a hardware mechanism used on the most elementary level to allow no other processor access to a block of memory while one processor is modifying a value in that block of memory (mutual exclusion). This hardware mechanism can be used in combination with a software mechanism (such as a semaphore) where arbitration between multiple processors using the system bus is desired.

System bus lock failures are detected by testing a software semaphore lock mechanism (which uses the hardware system bus lock mechanism) and its ability to successfully manage a "test" location in shared memory. The ability of the system bus lock used for all semaphores for the particular shared-memory resource is verified by testing this semaphore. Therefore, only one test semaphore is needed per shared-memory resource. However, the test semaphore does not need to be dedicated to diagnostics; instead, only one of the memory locations managed by the semaphore is dedicated to diagnostics. Furthermore, each processor that uses the system bus lock hardware mechanism as part of its software semaphore mechanism should conduct the same test of the same test semaphore and test memory location. Each processor writes a unique (for that processor) value to the test location.

A system bus lock fails in one of two ways: A processor is not allowed access in an acceptable amount of time, or more than one processor is allowed to simultaneously access a region. Semaphore lock failures most likely indicate failures in the hardware used to implement the non-interruptable read-modify-write instruction on which the lock depends.

Figure 5 presents a sample algorithm that is used to satisfy the requirements of the mutual exclusion diagnostic algorithm. The amount of time necessary to wait after writing the unique value to the test location depends on the asynchronous properties of the processors sharing the memory resource. If the processors are truly asynchronous, practice has shown that failures are detected even when this time is minimal.

```
REPEAT attempting to acquire the semaphore until either an unacceptable
    amount of time has expired, or the semaphore is acquired.
IF an unacceptable amount of time has expired, THEN
    REPORT the failure.
    EXIT the algorithm.
END IF
IF the semaphore became acquired, THEN
    WRITE a unique test pattern to a test location managed by the semaphore.
    WAIT some amount of time.
    READ back that value to determine if another processor was erroneously
        allowed to access to the location and corrupted it with its own
        unique value.
    IF the value read was not the value written, THEN
        REPORT the failure.
        EXIT the algorithm.
    END IF
END IF
REPORT no failure.
EXIT the algorithm.
```

Figure 5: Example Mutual Exclusion Diagnostic Algorithm

Unexpected Software Interrupt Diagnostic Algorithms

For environments that do not allow interrupts, the occurrence of an interrupt indicates a fatal failure. For environments that do allow interrupts, the occurrence of an interrupt that is not being used also indicates a fatal failure. The functions described in this section provide a default interrupt service for all interrupts. The algorithm used during system start-up time stores the address of the default servicing function in memory locations appropriate for each interrupt vector. Since these algorithms run on the 8086 class of microprocessors (including the 80286 in real mode only), there are 256 possible different interrupts. The 4-byte interrupt vectors are stored in the first 4 x 256 = 1024 bytes in memory with the vector for interrupt zero stored at location zero and each successive vector stored at contiguously higher locations.

If a service routine for a specific interrupt is needed, the address of that routine is copied over the address of the default service routine for that interrupt vector.

The default interrupt service routine determines which interrupt occurred, reports the interrupt failure, and halts the CPU.

Diagnostic Support Hardware

Westinghouse uses commercially-available microcomputer boards (IEEE-796 compatible) in many of its real-time systems. Each subsystem consists of several boards in a single chassis. The chassis provides physical support, power, and a status panel. Each of the boards serves one of three functions:

1. Host Processors - The host processors are the microcomputer boards which actually perform the application function. The processors may share the function or they may be arranged in a redundant configuration.
2. Slave Processors - The slave processors are intelligent I/O subsystems.
3. I/O Boards - These are non-intelligent interface boards.

Some of the hardware necessary to support diagnostic functions is not generally found on commercially-available boards. A general purpose support function card has been designed for use in real-time systems. This card, the Multibus Diagnostic Monitor (MDM), is an intelligent IEEE-796 slave processor. The on-board microprocessor communicates with the host processors via its shared memory.

The basic features of the MDM are as follows:

- Non-Volatile Memory - The MDM provides 2K bytes of IEEE-796 bus accessible non-volatile memory for the retention of diagnostic and post mortem information. This information is placed in the memory by the host processors. The information is accessible via a terminal driven by special maintenance software resident on the host processors.
- Temperature Monitoring - The MDM interfaces to as many as eight temperature sensors (Analog Devices AD590LH). Two of these sensors will be located on the card chassis itself. The other six sensors are intended to be distributed throughout the cabinet.

  The software running in the MDM reads these temperature sensors, converts the readings to degrees, and places the values into shared memory for use by the host processors. The software also compares the temperatures to programmable high and low thresholds. If any threshold is exceeded, an alarm indication is placed in shared memory and the alarm indicator on the status panel is lit.
- Power Supply Monitoring - The MDM monitors the IEEE-796 power supplies and the redundant 15-Vdc power supplies (used by signal conditioning modules associated with the I/O signals). Additionally, six channels of general purpose power supply monitoring are provided.

  The software running in the MDM reads these voltages, converts the readings to volts, and places the values into shared memory for use by the host processors. The software also compares the values to programmable high and low thresholds. If any threshold is exceeded, an alarm indication is placed in shared memory and the alarm indicator on the status panel is lit.
- Door Limit Switch Monitoring - The MDM interfaces to two door limit switch loops. Each loop consists of several door switches wired in series. Thus, if any door is opened the loop is broken.

  The software running in the MDM reads these contacts and places the values in shared memory for use by host processors.
- Subsystem Identification - Ten bits of digital input are available for use as a subsystem identification code. The software running in the MDM reads this code and places the value in shared memory for use by host processors. The host processors compare this value to a copy burnt into their PROM memory. In this manner, a host can determine if it is in the proper subsystem.
- Redundant Subsystem Select Logic - The MDM provides circuitry to perform the selection of a subsystem in a redundant subsystem architecture. The MDM boards in the two subsystems are cross coupled; this electrical interface is accomplished via the front edge connectors. The MDM drives two indicators on the local status panel. The "RUN" light indicates that the local subsystem is operational. The "CONTROL" light indicates that the local subsystem is actually in control of the process.
- Host Watch Dog Timer/Auto Restart - The host processors strobe individual keep-alive locations in the MDM's shared memory. If all of the host processors fail to do this, the MDM causes the redundant subsystem selection logic to pass control to the other subsystem. The MDM also asserts the IEEE-796 INIT line, holding the subsystem in reset. The MDM can optionally be programmed to release the reset line, thus restarting the subsystem. The Auto Restart option is disabled by default. Any host processor can enable this feature if desired. If the feature is enabled, the Auto Restart Enable indicator on the status panel is lit.
- MDM Hardware Deadman - A hardware deadman is provided for the MDM processor. Once enabled, if the deadman is not serviced, the MDM processor is reset.
- I/O Module AOK Loop - Several custom signal conditioning modules (E-Series modules) are associated with each subsystem. These modules are used to interface field signals to the IEEE-796 cards. They provide all of the signal conditioning, signal conversion, isolation, buffering, termination, and testability requirements of the subsystem. Each module creates an "All is OK" (AOK) contact closure. The normally-closed AOK contacts of all the modules associated with a given subsystem are tied in series. This loop is monitored by the MDM.

  The software running in the MDM reads this contact loop and places the value in shared memory for use by the host processors.
- Reset on +5 Vdc out of Specification - The chassis +5 Vdc supply is monitored. If it goes out of specification, the IEEE 796 INIT line is asserted.

- IEEE-796 Compatibility IEEE-796 compliance is Slave D8 M20 which means that the MDM is a slave board with an eight-bit data path and a twenty-bit address bus.

  Pins 1 through 40 of the P2 connector are defined as category number one signals (unconstrained use) and used for MDM specific functions. To prevent the use of other cards in slots wired for a MDM card, the P2 connector is keyed between pins 7 & 9 and pins 51 & 53.

- Status Panel Interface - The MDM provides the interface to the status panel located on the chassis. All status panel connections are made via the IEEE-796 P2 connector. Interfaces for the following switches and indicators (in addition to the those mentioned elsewhere) are provided:

  - Alarm Indicator
  - Reset Pushbutton

Conclusion

As microprocessor-based technology is applied to critical applications, the design of fault-tolerant systems becomes increasingly important. Low-level diagnostic software and hardware are critical components, but indiscriminate and excessive use of embedded diagnostics can diminish the overall performance of the system. A judicious combination of high-level, fault-tolerant architectures and low-level diagnostic hardware and software is a must. Westinghouse has achieved this ideal combination in the design of many critical systems.

Acknowledgements

The authors wish to acknowledge the following individuals for their contributions: G. W. Remley, B. M. Cook, J. A. Neuner, J. E. Hasenkopf, D. M. Rao, L. L. Santoline, and J. F. Sutherland.

Bibliography

[1] W. Barraclough, A. C. L. Chiang, and W. Sohl, "Techniques for Testing the Mircocomputer Family", Proceedings of the IEEE, Vol. 64, pp. 943-950 June 1976.

[2] J. Knaizuk, Jr., and C. R. P. Hartmann, "An Algorithm for Testing Random Access Memories", IEEE Transactions on Computers, Vol. C-26, pp. 414-416 April 1977.

[3] J. Knaizuk, Jr., and C. R. P. Hartmann, "An Optimal Algorithm for Testing Stuck-at Faults in Random Access Memories", IEEE Transactions on Computers, Vol. C-26, pp. 1141-1144 November 1977.

[4] R. Nair, S. M. Thatte, and J. A. Abraham, "Efficient Algorithm for Testing Semiconductor Random Access Memories", IEEE Transactions on Computers, Vol. C-27, pp. 572-576 June 1978.

[5] C. A. Papachristou and Narendar B. Sahgal, "An Improved Method for Detecting Functional Faults in Semiconductor Random Access Memories", IEEE Transactions on Computers, Vol. C-34, pp. 110-116 February 1985.

[6] K. K. Saluja and K. Kinoshita, "Test Pattern Generation for API Faults in RAM", IEEE Transactions on Computers, Vol. C-34, pp. 284-287 March 1985.

[7] V. P. Srini, "Fault Location in a Semiconductor Random-Access Memory Unit", IEEE Transactions on Computers, Vol. C-27, pp. 349-358 April 1978.

MULTIPROCESSOR SHARED-MEMORY INFORMATION EXCHANGE

L. L. Santoline, M. D. Bowers, and A. W. Crew

Engineering Technology Division
Westinghouse Electric Corporation
Research & Development Center
1310 Beulah Road
Pittsburgh, PA 15235
(412) 256-2537/2456/2539

C. I. Roslund and W. D. Ghrist III

Nuclear and Advanced Technology Division
Instrumentation Technology and Training Center
Westinghouse Electric Corporation
P.O. Box 598
Pittsburgh, PA 15230
(412) 733-6780/6343

Abstract

In distributed microprocessor-based instrumentation and control systems, the inter- and intra-subsystem communication requirements ultimately form the basis for the overall system architecture. This paper describes a software protocol which addresses the intra-subsystem communications problem. Specifically, the protocol allows for multiple processors to exchange information via a shared-memory interface. Our primary goal is to provide a reliable means for information to be exchanged between central application processor boards (masters) and dedicated function processor boards (slaves) in a single computer chassis. The resultant Multiprocessor Shared-Memory Information Exchange (MSMIE) protocol, a standard master-slave shared-memory interface suitable for use in nuclear safety systems, is designed to pass unidirectional buffers of information between the processors while providing a minimum, deterministic cycle time for this data exchange. This is achieved by providing multiple buffers for each unique block of information passed between the two processors. Another important feature of the design is that the interface between masters and slaves is identical for different types of slave processors. Thus, the amount of custom software in the final system is minimized. The use of standard system software not only eases initial software verification and validation requirements, it also simplifies long term system software maintenance.

Introduction

This paper describes the design of a standard shared-memory interface for intra-subsystem communications. The interface provides a method for reliable information exchange between processors in a single computer chassis which have access to a subsystem bus and shared-memory resources. This interface protocol, known as Multiprocessor Shared-Memory Information Exchange (MSMIE), is optimized for real-time critical process control and instrumentation systems such as nuclear safety systems.

A distributed processing architecture is a natural choice when designing critical real-time systems such as nuclear safety systems. By distributing the processing requirements of a time-critical function across multiple processors, the tasks to be performed by each component processor are reduced. Furthermore, the processing tasks of most subsystems within a system can be functionally viewed as a unique application function and a set of common "operating system" type functions such as I/O handling and pre-processing, external communication processing, and diagnostics, to name a few. By off-loading the dedicated "operating system" type functions onto individual "slave" processors, two distinct advantages are gained. First, the hardware and software for the processors performing the common system tasks may be of a standard, configurable design. Secondly, the processing burden of the subsystem application processor, or "master" processor, is substantially reduced, both in volume and execution time. However, the use of multiple processors to implement a single subsystem creates an additional communications burden: that of communications among the processors within the subsystem. The most efficient mechanism for intra-subsystem communication is a tightly-coupled architecture in which all processors share a bus, and communicate via a bus-accessible shared memory.

A typical architecture for a functionally-distributed computer system is shown in Figure 1. The system shown contains two subsystems, each containing a "master" processor, a "slave" processor, and a shared memory. Intra-subsystem communications are accomplished via the shared memory, and inter-subsystem communications are accomplished between the two slave processors via an unspecified physical communicating channel.

In order to avoid designing unique interfaces between master processors and each different type of slave processor, a standardized shared-memory interface protocol is required. The MSMIE protocol defines such an interface. It is optimized for real-time, process control type systems, such as nuclear safety systems, where operation in a non-interrupt driven environment is highly desirable.

The MSMIE Interface

The Participants

One of the primary goals of the MSMIE design is to identify a set of standard, configurable slave processor boards, and to design the slave processor software so that each slave microprocessor board of a given type could be used interchangeably throughout the overall system. This type of design not only minimizes the number of different microprocessor board types and the amount of custom software in the overall system, it also restricts custom software to the master processor boards. In a nuclear safety system, this type of design significantly improves overall system quality and integrity by focusing the total design effort (including verification and validation), on a small number of hardware and software components which are used as "building blocks" throughout the system. An additional benefit of such a standardized design is that long term hardware and software maintenance is simplified. For all of these reasons, the MSMIE interface has been designed so that master processors have the ability to configure individual slaves and thereby tailor them to the specific requirements of the subsystem. Thus, slave processors are dependent upon the master processors for their configuration information, which is passed to the slave processors during initialization. Slave processors communicate with the master processors via the subsystem shared memory, usually resident on each slave processor board. To further isolate the functionality of the slaves, they are only permitted to communicate with a master processor via the shared-memory interface. Slave-to-slave communications within a subsystem are not permitted except through some external communications device or via the subsystem master.

For some critical subsystems, an added degree of fault tolerance implemented via redundant subsystem master processors may be necessary. For true fault tolerance, the master processors must be fully redundant and isolated so that a fault of one master processor will not cause the others to fail. To allow the MSMIE interface to function properly, only one master processor may have the power to control the shared-memory interface at any given time. This processor is denoted the "primary" master, while all other masters are called "auxiliary" masters. The primary master is responsible for initial communications establishment with the slave processors, configuration of the slave processors, and if warranted, resetting the slave processors. The auxiliary masters may only monitor the shared-memory interface until after the slaves have been configured and MSMIE communications are fully established. At that point, the auxiliary masters may participate in shared-memory message passing to and from the slave processor boards.

Shared-Memory Organization

Master and slave processor communications is implemented via a predefined set of shared-memory data structures, which form the basis of the MSMIE interface. Between each slave processor and the subsystem host processors, a shared-memory region exists which is organized as shared-memory configuration data structures followed by message buffers. To maintain configurability of the slave processors, the shared-memory data structures are passed from the master to the slave processors during MSMIE initialization. The data structures contain information used by the slave to define the number and operation of any physical communications channels on the slave, the number, directionality, and definition of the messages communicated over each physical channel and between the masters and the slave, and other general configuration information. In addition, the data structures contain locations for passing diagnostic and run-time status between the masters and the slave, and locations for controlling the establishment of communications and resetting the slave from the primary master.

Method

The MSMIE protocol is designed so that each message buffer exchanged between the slave and masters is unidirectional, with the contents of the message being a continuously-updated image. The method for one processor to communicate with another is:

- The processor sending information will continually copy the newest image of a message into a shared-memory buffer
- The processor receiving information will read from the shared-memory buffer containing the newest image.

A shared-memory buffer is either updated by a master processor, and the flow of information is from master-to-slave, or alternatively, a shared-memory buffer is updated by a slave processor, and the flow of information is from slave-to-master.

The use of shared memory as a means of exchanging information between multiple, asynchronous processors is only successful if a mechanism exists to prevent simultaneous access to a given memory resource. Without this mechanism, the possibility of "data tearing" arises. Data tearing occurs when one processor writes to a memory area while it is being read by another processor. If this situation exists, it is possible that the processor reading the memory area actually reads portions of both old and new data. This can happen whenever the memory location being accessed is of a size that requires multiple machine instructions to read or write the location. Consider the following simple example, where processor number one reads a location which is concurrently being written to by processor number two:

WORD VALUE			
High Byte	Low Byte	Processor One	Processor Two
55	55	READ Low = 55	
55	33		WRITE Low = 33
33	33		WRITE High = 33
33	33	READ High = 33	

As indicated, the word value read by processor number one is invalid as it contains the low byte of the "old" data value, but the high byte of the "new" data value. This simple example can be extended to more sophisticated situations, where the integrity of whole blocks of data must be maintained.

In order to prevent data tearing, mutually-exclusive access to the shared-memory area must be guaranteed. In MSMIE, this is partially accomplished with software semaphores. The semaphores allow a processor, while it has access to a particular shared-memory area, to prevent, or "lock out", other processors from accessing that same shared-memory area.

While semaphores prevent data tearing, they introduce the possibility of one processor being denied access to a shared-memory area because that area is locked by another processor. The processor which desires buffer access must wait for the other processor to release the locked shared memory area. This

is readily apparent if only a single shared-memory buffer is allocated to hold each message image. In this case, the buffer acquisition/release scheme of master and slave processors is as shown in Figure 2. As illustrated, buffer contention problems are normal with a single buffer per message allocation scheme. When messages are passed from "slave-to-master", the master processor must wait for the slave to update the data in the buffer, and then release the buffer so the master can access the newest data. While the master is reading the new data from the buffer, the slave has no free area to build a new message. For message images passed from "master-to-slave," the situation is reversed. In either case, each processor must wait for the buffer to be in the correct state (either "idle" or "newest") before it may access the buffer. There is no guarantee that the buffer will be in the correct state at any given time. When this primitive message passing interface is used for information exchange between the master and slave, only a fraction of the full power of a multiple processor architecture can be realized, since each processor wastes a portion of its execution cycle waiting for the other processor to release the shared-memory resource.

The addition of a second buffer for each message image communicated between the master and slave processors solves some buffer contention problems. While one or more processors are reading a message image from the first shared-memory buffer, another processor may be building a message update in the second shared-memory buffer. Using this method, the participating processors can simultaneously operate on (read from or write into) shared-memory buffers without interfering with one another.

However, because master and slave processors may run asynchronously, buffer contention is still possible even with dual memory buffers allocated for each message image. This is true because there are no real restrictions on the amount of time a processor can hold a buffer assigned to itself. The buffer contention problem which occurs using dual shared-memory buffers for each message image is explained in the following scenario (see Figure 3). Assume the slave processor is operating with a slower cycle time than the master processor. This implies that the slave will take longer to access and release a data buffer than the master. When the slave processor is reading an image of a message from one shared-memory buffer, the faster master processor builds a new image of the message and places it in the second shared-memory buffer. If the master processor builds another new message image before the slower slave processor has finished using its data buffer, then the master processor must either wait for the slave to release its buffer to the idle state, or overwrite the previous "newest" image with the fresher data. The first alternative is undesirable because the operation of the two processors is coupled. The second alternative is also undesirable because the newest image is destroyed. In the second case, when the slave processor finally releases its buffer, it can no longer immediately access a new image. It must instead wait for the master processor to finish updating the newest image and release the buffer to the newest state.

To eliminate the possibility of buffer contention, three shared-memory buffers can be allocated for each message image. With this scheme, at any given time, one of the buffers can hold the newest complete image, a second buffer can be assigned to a processor for reading an image, while a third buffer can be assigned to a processor for building another new image. A buffer is guaranteed to be available to each processor at the time it requests access to a buffer, and a third buffer is always available to prevent the newest complete image from being destroyed.

Thus, for each data image which must be communicated between a slave and the masters' processors, the MSMIE protocol maintains a set of triplicated buffers. Information about the buffers and access control to the buffers is provided by "buffer descriptor tables" in shared memory. Each set of triplicated buffers has an associated buffer descriptor containing the following entries:

- A three-element Buffer_Status array which holds the status of each of the three shared-memory buffers. Each buffer can have one of five buffer statuses: "idle", "assigned to master", "assigned to slave", "newest", and "not used".
- A semaphore location which provides exclusive access to the Buffer_Status array for both master and slave processors. The semaphore and the Buffer_Status array are used to control access to the triplicated shared-memory buffers.
- A Number_of_Readers location which maintains a count of the number of master processors currently reading the buffer whose status is "assigned to master".
- An Access_Mode location which determines the number of read accesses permitted to "newest" data buffers.

The method in which the buffer descriptor parameters are used varies according to the direction of information flow and whether the processor desiring buffer access is a master or slave processor. The use of the buffer descriptor parameters will be described in the following sections.

Slave-to-Master Message Passing, Slave Processor Buffer Acquisition/Release: In slave-to-master message passing, the slave processor marks messages for use by the master processor. Typically, the slave processor has a buffer assigned to it at initialization to hold the first message. Then, when the acquire/release buffer procedure is invoked, the slave releases the buffer which was assigned to it (by updating its status to "newest"), then acquires an "idle" buffer to hold the next update of the message. The procedure which the slave processor follows to release its current buffer and acquire an "idle" buffer is described below:

1. The slave processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore. Once the slave processor has the semaphore in the locked state, the master processor is denied access to the Buffer_Status array.
2. The Buffer_Status array is searched for a status of "newest". If a "newest" status is found, then the data which the slave processor is updating for the master replaces this buffer, so the Buffer_Status of "newest" is changed to "idle".
3. The Buffer_Status array is searched for a status of "assigned to slave". If the status of "assigned to slave" is not found, then an error has occurred.
4. The buffer whose status is "assigned to slave" is changed to "newest". This completes the release of the "assigned to slave" buffer. An "idle" buffer must now be acquired.
5. The Buffer_Status array is searched for a status of "idle". This buffer will be used to hold the next message update. If an "idle" buffer is not found, then an error has occurred.
6. The slave processor acquires the "idle" buffer by

changing its status from "idle" to "assigned to slave".

7. The buffer descriptor semaphore is released.

Slave-to-Master Message Passing, Master Processor Buffer Acquisition/Release: The master processor accesses the newest messages provided by the slave processor. Two separate procedures are provided: one to acquire the "newest" buffer, and a second procedure to release the "assigned to master" buffer once the data has been used. The actions which must be taken by the master processor in order to acquire "newest" data buffers are described below:

1. The master processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore. Once the master processor has the semaphore in the locked state, the slave processor is denied access to the Buffer_Status array.
2. The Buffer_Status array is searched for a status of "assigned to master".
3. If an "assigned to master" buffer is found, then another master processor has acquired this buffer. (In this case, multiple master processors have access to the slave's shared memory.) The master processor presently desiring buffer access is constrained to read the data in the current "assigned to master" buffer.
4. If an "assigned to master" buffer is not found, then this master processor is free to search the Buffer_Status array for a "newest" buffer. If a "newest" buffer is found, then it is acquired by changing the Buffer_Status to "assigned to master". If a "newest" buffer is not found, then no message has been provided by the slave processor.
5. To mark the number of master processors reading this buffer, the Number_of_Readers location is incremented.
6. The buffer descriptor semaphore is released.

At this point, the master processor uses the data from the "assigned to master" buffer. When the master no longer desires access to this buffer, it may release the "assigned to master" buffer such that the slave processor can reuse this buffer. The procedure which the master follows to release the buffer is described below:

1. The master processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore. Once the master processor has the semaphore in the locked state, the slave processor is denied access to the Buffer_Status array.
2. The Number_Of_Readers location in the buffer descriptor is decremented, as this master processor no longer requires access to the "assigned to master" buffer.
3. If the Number_Of_Readers location is not equal to zero, then another processor has access to the "assigned to master" buffer. If this is the case, the "assigned to master" buffer cannot be released. The Buffer_Status is left in the "assigned to master" state, and the buffer descriptor semaphore is released.
4. If the Number_Of_Readers location is now equal to zero, then this master processor was the only master processor using the "assigned to master" buffer, and the buffer may be released. In order to release the buffer, the Buffer_Status array is searched for a status of "newest". If such a buffer is found, or the Access_Mode location in the buffer descriptor indicates that each buffer is to be accessed only once, then the status of the buffer to be released is changed to "idle". Otherwise, the buffer which is to be released still contains the newest data, and it should be released by changing its status back to "newest".[*]
5. The buffer descriptor semaphore is then released.

Master-to-Slave Message Passing, Master Processor Buffer Acquisition/Release: In master-to-slave message passing, the master processor marks messages for use by the slave processor. The master processor uses two separate procedures to access the shared-memory data buffers; one to acquire an "idle" buffer, and a second procedure to release the "assigned to master" buffer once the new message has been moved into the shared-memory buffer. The actions which must be taken by the master processor to acquire an "idle" buffer are described below:

1. The master processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore.
2. The Buffer_Status array is searched for a status of "idle". If an "idle" buffer is not found, then an error has occurred.
3. The buffer whose status is "idle" is acquired by the master processor by changing the status to "assigned to master".
4. The buffer descriptor semaphore is released.

At this point, the master processor moves the "newest" message image into the acquired shared-memory data buffer. Once this data transfer is complete, the master processor may release the "assigned to master" buffer such that the slave processor can use the "newest" data. The procedure which the master follows to release the buffer is described below:

1. The master processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore.
2. The Buffer_Status array is searched for a status of "newest". If a "newest" status is found, then the data which is being provided by the master processor replaces this buffer, so the Buffer_Status of "newest" is changed to "idle".
3. The Buffer_Status array is searched for a status of "assigned to master". This is the buffer which has been filled with the "newest" data. If such a buffer is not found, an error has occurred.
4. The buffer whose status is "assigned to master" is changed to "newest".
5. The buffer descriptor semaphore is released.

Master-to-Slave Message Passing, Slave Processor Buffer Acquisition/Release: The slave processor must access the newest message provided by the master processor. Two separate procedures are provided: one to acquire the "newest" buffer, and a second procedure to release the "assigned to

[*] If multiple master processors are used, the Access_Mode must be configured to allow an unlimited number of accesses to a single "newest" buffer. This is so all master processors are guaranteed access to a "newest" message once one has been provided by the slave.

slave" buffer once the newest data has been used by the slave processor. The actions which must be taken by the slave processor in order to access "newest" data buffers are described below:

1. The slave processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore.
2. The Buffer_Status array is searched for a status of "newest".
3. If a "newest" buffer is found, it should be acquired for use by the slave processor by changing its status to "assigned to slave". If a "newest" buffer is not found, then the master processor has not yet provided any messages in shared memory.
4. The buffer descriptor semaphore is released.

At this point, the slave processor is free to use the data in the buffer assigned to it. When the slave processor no longer requires access to the "assigned to slave" buffer, it must be released so that the master processor can reuse the buffer. The procedure which the slave follows to release the buffer is described below:

1. The slave processor locks the Buffer_Status array by acquiring the buffer descriptor semaphore.
2. The Buffer_Status array is searched for a status of "newest". If such a buffer is found, or if the Access_Mode location in the buffer descriptor indicates that each buffer is to be accessed only once, then the status of the buffer to be released is changed to "idle". Otherwise, the buffer which is to be released still contains the "newest" data, and it should be released by changing its status from "assigned to slave" back to "newest".
3. The buffer descriptor semaphore is then released.

Multiple Channel Slaves: The basic method of applying triple buffering to shared-memory communications of data images has been described. This method can be extended to suit the particular needs of different types of slave processor boards. As previously mentioned, the slave processors are typically designed to offload the master processors from performing standard system tasks. One common slave processor function is simplex point-to-point (datalink) communications. This type of slave processor benefits from a variation of simple shared-memory triple buffering due to multiple communication channel considerations.

A datalink controller type of slave processor generally has greater than one physical communications device. Each of the physical communications channels (datalinks) can operate as a transmitter, receiver, or bidirectional channel. It is also possible that multiple message images are to be communicated over a single physical channel.

Because datalink activity is serial, only one message at a time can be transmitted or received on any given channel. Thus, triple buffering is implemented as follows: The number of shared-memory buffers required for each datalink channel is equal to two times the number of unique messages communicated over that channel plus one additional buffer. The "extra" buffer is for the physical channel itself, i.e., the buffer into which messages are received or from which messages are transmitted. In this arrangement, the shared-memory buffers are in a free pool of buffer space, and are not associated with a particular buffer descriptor except at initialization. At initialization, two shared-memory buffers are assigned to each buffer descriptor, and are initialized to the "idle" state. The third Buffer_Status in each buffer descriptor is initialized to the "assigned to slave" state, as the third buffer for all buffer descriptors associated with a single channel corresponds to the single "extra" buffer which is assigned to the physical channel. In this case, the "assigned to slave" Buffer_Status can be thought of as "assigned to physical channel". Because the buffers are not rigidly allocated to a particular buffer descriptor, the size of each of the allocated buffers must be at least as large as the largest message received or transmitted over the given channel. At any given time, two buffers are associated with each particular message image, and the third buffer is always assigned to the datalink controller physical communications device. When triple buffering is implemented in this manner, the method of acquiring and releasing buffers from the master side is identical to that previously described. From the datalink controller side, buffer acquisition and release is a "swapping" process.

On datalink controller receive channels, messages are received over the datalink and must be marked for use by the master processor. The messages are received into the shared-memory buffer assigned to the physical channel. Once a new message has been received, the datalink controller must determine which buffer descriptor the message is associated with, and find the correct buffer descriptor. This buffer descriptor is where the "assigned to slave" buffer must be returned, and from where an "idle" buffer must be acquired to rearm the physical channel. Once the correct buffer descriptor is found, the procedure which the datalink controller follows to release its current buffer and acquire an "idle" buffer is identical to standard triple buffering for the slave-to-master message passing case described previously.

For transmitter channels, the master processor provides "newest" message buffers which contain data to be transmitted over the physical datalinks by the datalink controller slave. If multiple messages are transmitted on a single datalink, each message is transmitted separately and in order. In this case, buffer acquisition and release is a two step buffer swapping process, as described below:

1. Acquiring buffers for transmission involves swapping a current "assigned to slave" buffer with the "newest" buffer from the buffer descriptor containing the least recently-transmitted message. "Newest" buffer acquisition is described below:
   a. The semaphore of the buffer descriptor corresponding to the least recently-transmitted message is acquired.
   b. The Buffer_Status array is searched for a status of "newest". If a "newest" status is found, then the datalink controller must swap the current "assigned to slave" buffer with the "newest" buffer so that the datalink controller can transmit the newest data.
   c. The Buffer_Status of the "assigned to slave" buffer is changed to "idle".
   d. The Buffer_Status of the "newest" buffer is changed to "assigned to slave".
   e. The buffer descriptor semaphore is released.
2. The transmission of the "assigned to slave" buffer is initiated. Upon the completion of transmission, the buffer must be returned so that the master processor can reuse the buffer. The buffer must be returned to the same buffer descriptor from which

it was acquired. The procedure for returning "assigned to slave" buffers is described below:

a. The buffer descriptor semaphore of the most recently-transmitted buffer is acquired.

b. The Buffer_Status array is searched for a status of "newest". If such a buffer is found, or if the Access_Mode of this buffer descriptor indicates that each "newest" buffer is to be used only once, then the buffer which is presently "assigned to slave" remains in that state. Otherwise, the buffer which is "assigned to slave" still contains the "newest" data, and it should be released by changing its status from "assigned to slave" back to "newest".

c. If the "assigned to slave" buffer was released (its status changed to "newest"), then an "idle" buffer must be acquired for use by the datalink controller. The Buffer_Status array is searched for a status of "idle". This buffer is acquired by changing its status to "assigned to slave".

d. The buffer descriptor semaphore is released.

The triple buffering procedure followed by slave processors which are similar to datalink controllers is just an extended version of simple shared-memory triple buffering. However, this method significantly reduces the memory requirements when many messages of a similar size must be transmitted or received over a single physical channel. This method reduces to simple triple buffering when only one message is transmitted or received per physical channel.

Multiple Master Processor Considerations: The basic triple buffer acquire/release algorithms which have been described are applicable regardless of whether one or more than one master processor is communicating with the slave processor. When multiple masters are present, the Number_of_Readers location in each buffer descriptor allows each master processor to simultaneously read the same buffer of a message passed from slave-to-master. However, only a single master processor is permitted to supply each image of a message passed from master-to-slave in order to prevent master-to-master buffer contention.

Although no changes are required to the buffer acquire/release algorithms, there are various operating constraints when more than one master processor is present in a subsystem:

1. Because each multiple master processor may run asynchronously with respect to other master processors and the slave processor, one master could essentially prevent the others from ever reading data provided by the slave if the Access_Mode selection on slave-to-master message passing buffers were configured for one time buffer access. This leads to the requirement that all buffer descriptors with direction "slave-to-master" must be configured to allow an unlimited number of accesses to a single "newest" buffer to ensure that all master processors are able to access the "newest" data at least once per master processor cycle.

2. A timing constraint must be placed on the amount of time any master is allowed to access a buffer. This constraint is necessary so that a buffer which is "assigned to master" is guaranteed to be released by all masters at least once per master processing cycle. For a system with M masters, the amount of time any master processor may assign a buffer to itself must be less than $1/M$ of the smallest loop cycle time of any master.

Reliability

The method of using triplicated buffers for message passing between master and slave processors provides for information exchange between the processors within a minimal and deterministic time frame. For use in nuclear safety systems, the information exchange must also be extremely reliable. The MSMIE protocol provides reliability in the data exchange by several mechanisms, which are summarized below:

- A field specifying the length of the message is embedded into a header which is part of every message image.
- A message serial number is embedded into the message header. The serial number is used by the receiving processor to determine if a message has been read before.
- New message buffers are timestamped when they are placed into the shared memory. The receiving processor can calculate the age of a buffer by comparing the timestamp of the buffer with a representation of "current time", maintained in shared memory by subsystem slaves.
- Source-to-destination error detection is provided by a word-wide checksum which is embedded into the message. The checksum is computed by the processor which originates a message; it is recomputed by the end processors which eventually receives the message.

Summary

The MSMIE protocol has several features which make it ideally suited to inter-processor communications in distributed, microprocessor-based nuclear safety systems. At this time, implementation of the MSMIE protocol is a central part of the embedded software of several large Westinghouse nuclear system designs. The MSMIE protocol maximizes overall system performance in a multiprocessor environment while guaranteeing reliable communications between processors, deterministic performance, and maximum software reusability. This protocol represents a significant development in the design of nuclear safety system software.

Acknowledgments

The concepts and procedures described in this paper are taken from internal Westinghouse documents authored by the following persons: Mark D. Bowers, Albert W. Crew, William D. Christ III, Gilbert W. Remley, Charles J. Roslund, and Linda L. Santoline.

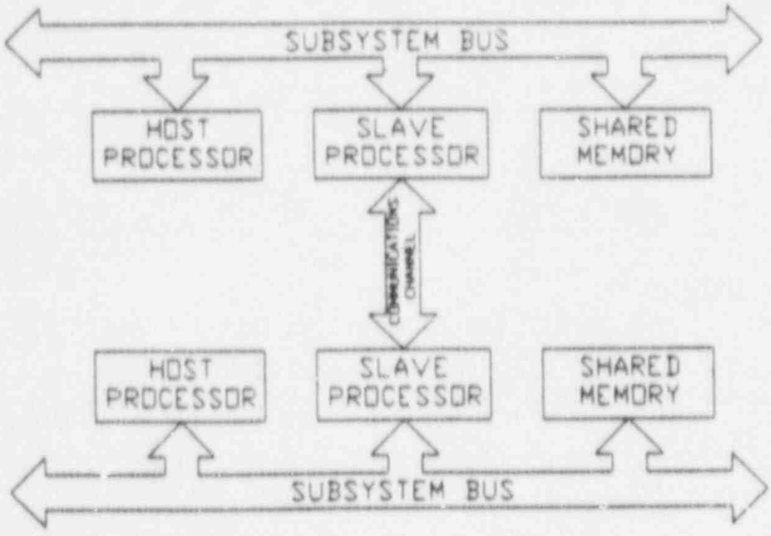


Figure 1: Intra-Subsystem and Inter-Subsystem Communications

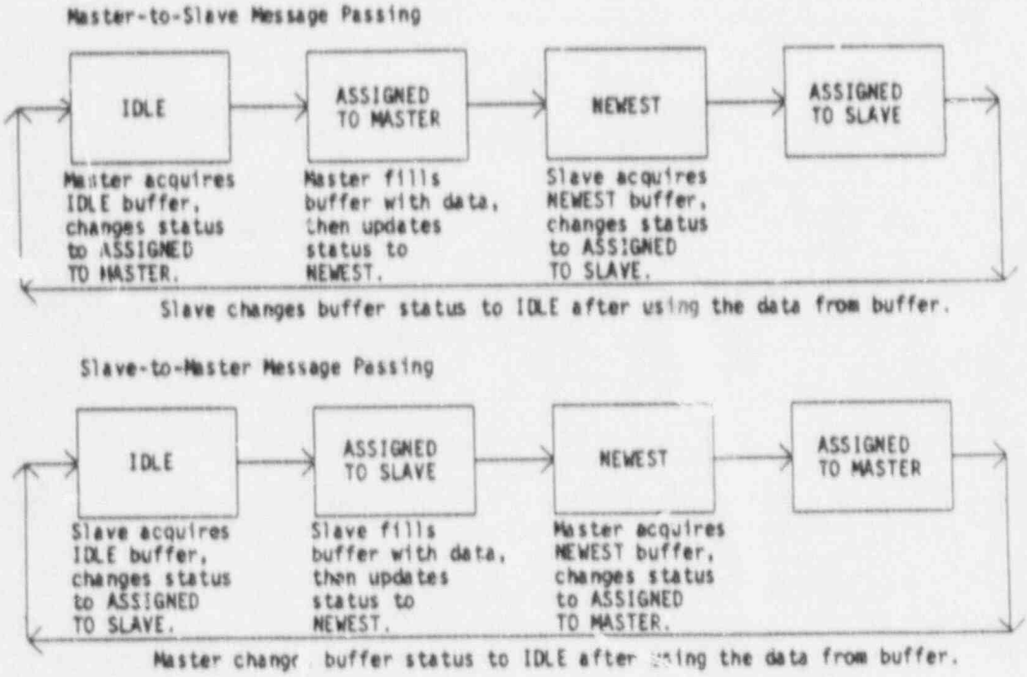


Figure 2: Master/Slave Buffer Acquisition and Release -- Single Buffer Per Message Case

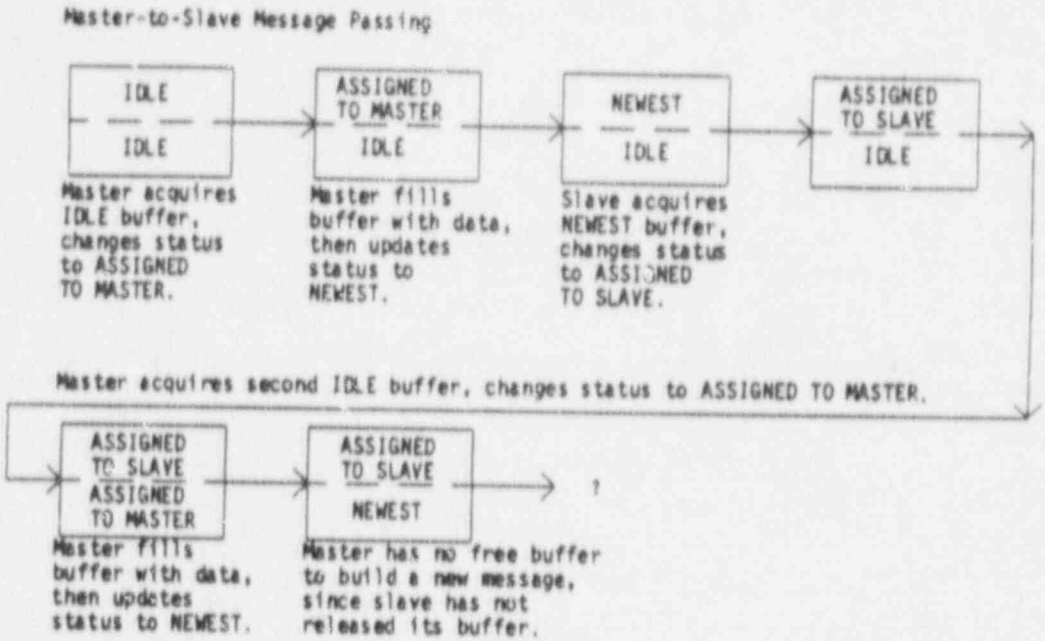


Figure 3: Master/Slave Buffer Acquisition and Release -- Dual Buffers Per Message Case

(4)

RESEARCH ACTIVITIES

CURRENT AND FUTURE PROGRAMS ON THE USE OF DIGITAL COMPUTERS IN NUCLEAR POWER PLANTS

* LESSONS LEARNED: EXPERIENCE IN OTHER COUNTRIES

* REVIEW CRITERIA - HUMAN FACTORS ASPECTS - ADVANCED I&C

* COMPUTER CLASSIFICATION

* EXPERT SYSTEM VERIFICATION AND VALIDATION

* HALDEN REACTOR PROJECT PROGRAMS
  SOFTWARE TOOLS
  SOFTWARE TEST AND EVALUATION

* CLASS 1E DIGITAL COMPUTER SYSTEMS

NUREG/CR-5348, MAN-MACHINE INTERFACE ISSUES IN NUCLEAR POWER PLANTS

LESSONS LEARNED: EXPERIENCE IN OTHER COUNTRIES

CANADA

DARLINGTON: REVERSE ENGINEERING

BRUCE: QUALITY CONTROL

FRANCE

N4 SERIES: FRONT END REQUIREMENTS ANALYSIS

GERMANY

KWU: 10 YEAR PROGRAM, USE OF CASE TOOL

TUV NORDDEUTSCHLAND: SOSAT

PROJECT: REVIEW CRITERIA - HUMAN FACTORS ASPECTS - ADVANCED I & C

CONTRACTOR: OAK RIDGE NATIONAL LABORATORY

OVERALL OBJECTIVE: TO DEVELOP REGULATORY REVIEW CRITERIA FOR USE IN EVALUATING THE SAFETY IMPLICATIONS OF HUMAN FACTORS ASSOCIATED WITH CURRENT PLANTS USE OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS AND WITH ADVANCED CONTROLS AND INSTRUMENTATION

INITIAL OBJECTIVE: PERFORM INDUSTRY SURVEY, DEFINE ISSUES

NUREG/CR-5439, JUNE 1990

HIGH RATED I&C ISSUES

WILL HIGH RESOURCE REQUIREMENTS OF VERIFICATION AND VALIDATION OF ADVANCED I&C DENY ANTICIPATED IMPROVEMENTS?

WHAT ARE THE CONFIGURATION CONTROL REQUIREMENTS FOR DIGITAL SYSTEMS BACKFITTED INTO NUCLEAR POWER PLANTS?

MEDIUM RATED I&C FACTS, ISSUES

ACCEPTANCE CRITERIA FOR ADVANCED I&C ARE NEEDED TO AVOID ITS PREMATURE USE AND POSSIBLE ERRORS

USE OF ONE-WAY (OUTWARD) COMMUNICATION WITH SAFETY SYSTEMS ENHANCES SECURITY

PROJECT: COMPUTER CLASSIFICATION

CONTRACTOR: OAK RIDGE NATIONAL LABORATORY

OBJECTIVE: REVIEW AND EVALUATE ADEQUACY OF EXISTING REGULATORY GUIDANCE FOR COMPUTER-BASED SAFETY SYSTEMS; WHERE NECESSARY, RECOMMEND DEVELOPMENT OF NEW GUIDANCE

A TWO YEAR PROGRAM INITIATED IN FEBRUARY 1989

DRAFT NUREG/CR UNDER REVIEW

PROJECT: EXPERT SYSTEM VERIFICATION AND VALIDATION GUIDELINES

CONTRACTOR: SCIENCE APPLICATION INTERNATIONAL CORPORATION

OBJECTIVE: TO DEVELOP AND DOCUMENT GUIDELINES FOR VERIFYING AND VALIDATING EXPERT SYSTEMS

DESCRIPTION: A JOINT RESEARCH PROJECT FUNDED BY EPRI AND NRC

A TWO YEAR PROGRAM INITIATED IN OCTOBER 1990

HALDEN PROJECT: SOFTWARE TOOLS

SOSAT - A SET OF TOOLS FOR SOFTWARE SAFETY ASSESSMENT

- DEVELOPED BY HALDEN FOR TUV NORDDEUTSCHLAND

- FUNCTIONS

  METRIC COMPUTATIONS

  STATIC ANALYSIS OF CODES

  DYNAMIC ANALYSIS

  SYMBOLIC EXECUTION

    PLAN TO DEVELOP ANALYSIS MODULES TO COMPARE PROGRAM WITH SPEC

  TIME ANALYSIS

- NOW IN USE BY TUV NORDDEUTSCHLAND FOR SAFETY EVALUATION OF MICRO PROCESSOR BASED SAFETY SYSTEM

HALDEN PROJECT: SOFTWARE TEST AND EVALUATION

GOAL: INCREASED SOFTWARE RELIABILITY

ONE SAFETY SYSTEM SPEC BY SAFETY AND RELIABILITY DIRECTORATE (UK)

INDEPENDENTLY CODED BY THREE TEAMS:

HALDEN REACTOR PROJECT
TECHNICAL RESEARCH CENTER OF FINLAND
CENTRAL ELECTRICITY RESEARCH CENTER (UK)

SCOPE OF RESEARCH:

FAULT FINDING STRATEGIES
TEST DATA SELECTION

HALDEN PROJECT: SOFTWARE TEST AND EVALUATION (CONT'D)

TEST DATA TYPES:

DETERMINISTIC DATA:

SYSTEMATIC DATA - MANUALLY PRODUCED TEST SPEC FUNCTIONS
PLANT SIMULATION DATA

RANDOM DATA:

UNIFORM DISTRIBUTION, EQUAL PROBABILITY INSIDE DATA RANGE
GAUSSIAN DISTRIBUTION, MEAN IN MID-RANGE
UNIFORM DISTRIBUTION AT BOUNDARIES
GAUSSIAN DISTRIBUTION AT BOUNDARIES

HALDEN PROJECT: SOFTWARE TEST AND EVALUATION (CONT'D)

TEST DATA EFFICIENCY:

FAULT DETECTION:
EACH PROGRAM SEEDED WITH 62 FAULTS
TEST EACH PROGRAM BY INPUT DATA TYPE
ALL FAULTS FOUND, MULTIPLE DATA TYPES REQUIRED

RESULTS:
MOST EFFICIENT:
UNIFORM DISTRIBUTION, INSIDE DATA RANGE
GAUSSIAN DISTRIBUTION, BOUNDARY
LEAST EFFICIENT:
GAUSSIAN DISTRIBUTION, MEAN IN MID-RANGE
SYSTEMATIC DATA

PROJECT: CLASS 1E DIGITAL COMPUTER SYSTEMS

CONTRACTOR: RADC\SOHAR INCORPORATED

OBJECTIVE: CONDUCT INDUSTRY SURVEY AND DEVELOP THE TECHNICAL BASIS FOR REGULATORY GUIDANCE ON THE DESIGN, DEVELOPMENT, TEST, AND ACCEPTANCE OF CLASS 1E COMPUTER SYSTEMS

A ONE YEAR PROGRAM

RESPONDS TO SPECIFIC USER REQUESTS FROM NRR

DEVELOP A REGULATORY GUIDE ON DESIGN AND DEVELOPMENT OF CLASS 1E COMPUTER SYSTEMS USING SURVEY AND RESEARCH RESULTS

COMPUTERS
IN
NUCLEAR POWER PLANTS

- PRESENT ACTIVITIES
  JOE JOYCE NRR/ICSB

- FUTURE APPLICATIONS
  JIM STEWART NRR/ICSB

- SOFTWARE V&V
  RAY ETS SMARTWARE ASSOC.

- STATUS OF EDF P20 SYSTEM
  JOHN GALLAGHER NRR/ICSB

EARLY DESIGNS

(1975 - 1980)

CORE PROTECTION CALCULATOR (CPC)

- FIRST NRC REVIEW OF A DIGITAL SAFETY SYSTEM THAT USES COMPLEX ALGORITHMS
- 6 MINICOMPUTERS
- 10 ANALOG & 2 DIGITAL TRIPS
- VENDOR/LICENSEE DESIGN AND INSTALLATION REQUIRED OVER 100 MAN YEARS
- NRC REVIEW EFFORT REQUIRED 18 MAN YEARS
  - MAJOR REDESIGN REQUIRED
  - STAFF DEVELOPED 27 POSITIONS

EARLY DESIGNS
(CONTINUED)

RPS-II/B&W

- 4 MICROCOMPUTERS
- 7 ANALOG & 3 DIGITAL TRIPS
- RESULTS
  - 14 ERRORS FOUND DURING INTEGRATION TESTING, INDICATING LACK OF DETAILED REVIEW PRIOR TO TESTING
  - STAFF/BOEING PERFORMED SNEAK ANALYSIS
    - 9 SOFTWARE DOCUMENT ERRORS
    - SOFTWARE CONTAINED NO SNEAK CONDITIONS
- STAFF REVIEW TERMINATED WHEN BELEFONTE CANCELLED

EARLY DESIGNS
(CONTINUED)

WESTINGHOUSE RESAR-414 INTEGRATED PROTECTION SYSTEM

- Major Change In Westinghouse Design
- Microcomputer-Based System Encompassing
  - RPS
  - ESFAS
  - Control Systems
- Review Group Formed To Assess Defense-In-Depth And Diversity Of The IPS
- Staff Developed Simplified Approach
  - Assess only system architecture
- Developed Block Concept And A Set Of Guidelines
- NUREG-0493, March, 1979
- Staff Issued Preliminary Design Approval On 11/78 With 9 Open Items

EARLY DESIGNS
(CONTINUED)

LESSONS LEARNED

- NRC Searched For Other Means To Reduce Regulatory Resource Expenditures

- Industry Should Perform Verification & Validation (V&V)

- Use ANSI/IEEE-ANS 7-4.3.2-1982

- Endorsed With RG 1.152

- NUREG-0493 D-I-D And Diversity

TYPES OF UPGRADES

- Direct Replacement Of A Single Analog Function With A Digital Equivalent
- Combining Several Analog Process Steps Into A Single Microprocessor
- Partial Replacement Of An Analog System With A Digital System
- Complete Replacement Of An Analog System With A Digital System
- Addition Of Digital Systems That Interface With Plant
- Replacement Of Mini-Computers With Microcomputers

PRESENT DESIGNS

PLANT	VENDOR/SYSTEM	SER
South Texas	Westinghouse Qualified Display Processing System	5/87
Vogtle	Westinghouse Plant Safety Monitoring System	6/87
Palisades	Gammametrics Thermal Margin Monitor	10/88
McClellan AFB	General Atomics TRIGA Digital Control Console	10/88
SONGS 1	Westinghouse NIS	12/88
Beaver Valley	Westinghouse Plant Safety Monitoring System	4/89
Big Rock Point	General Electric Neutron Flux System	4/89

PRESENT DESIGNS

(CONTINUED)

PLANT	VENDOR/SYSTEM	SER
WATTS BAR	WESTINGHOUSE EAGLE 21 RTD BYPASS	5/89
ARMED FORCES RADBI INST	GENERAL ATOMICS TRIGA DIGITAL CONTROL CONSOLE	7/89
GA TEST REACTOR	GENERAL ATOMICS TRIGA DIGITAL CONTROL CONSOLE	8/89
PRAIRIE ISLAND	WESTINGHOUSE DIGITAL FW CONTROL DIGITAL MEDIAN SIGNAL SELECTOR	1/90
ANO-2	COMBUSTION ENGINEERING UPGRADED CPC FROM MINI TO MICRO	1/90
HADDEM NECK	FOXBORO PHASE I RPS UPGRADE 2/90	3/90
DIABLO CANYON	WESTINGHOUSE DIGITAL MEDIAN SIGNAL SELECTOR	3/90
SEQUOYAH	WESTINGHOUSE EAGLE 21 REPLACES RPS EXCEPT NEUTRON FLUX	3/90

PRESENT DESIGNS

(CONTINUED)

PLANT	VENDOR/SYSTEM	SER
HADDEM NECK	FOXBORO PHASE II UPGRADE	4/90
HADDEM NECK	GAMMAMETRICS NIS UPGRADE	4/90
MAINE YANKEE	FOXBORO PRIMARY INVENTORY TRACKING SYSTEM	1/91
TOPICAL REPORT	GENERAL ELECTRIC WIDE RANGE NEUTRON MONITORING SYSTEM	10/90
PEACH BOTTOM	FOXBORO RPS UPGRADE	
TROJAN	WESTINGHOUSE REMOTE SHUTDOWN SYSTEM (NON-SAFETY)	
TURKEY POINT 3&4	WESTINGHOUSE EAGLE 21 TAS FOR RTD BYPASS	2/91

RESEARCH REQUEST (14 ITEMS)

- DEVELOP DIGITAL STANDARD (IEEE 279)
  - SOFTWARE/FIRMWARE
  - DATA COMMUNICATION
  - SECURITY
  - RELIABILITY
  - DIVERSITY
- DEVELOP ACCEPTANCE CRITERIA
  - FAULT TOLERANCE
  - FAULT AVOIDANCE
  - SLOW DEGRADATION
  - SOFTWARE TESTING
  - CONFIGURATION MANAGEMENT
- SURVEY INDUSTRY FOR CRITERIA
  - TESTING CRITICAL SOFTWARE
  - REVERSE ENGINEERING
  - FORMAL SPECIFICATION
  - SOFTWARE AUDIT TOOLS
- DEVELOP STANDARDS/CRITERIA
  - EXPERT SYSTEMS
  - ARTIFICIAL INTELLIGENCE

FUTURE APPLICATIONS

- EPRI ALWR (EVOLUTIONARY)
- GENERAL ELECTRIC ABWR
- COMBUSTION ENGINEERING SYSTEM 80+
- EPRI ALWR (PASSIVE)
- WESTINGHOUSE AP600
- GENERAL ELECTRIC SBWR
- COMBUSTION ENGINEERING SIR
- ABB/CE PIUS
- MHTGR/CANDU/.....
- RETROFITS AND UPGRADES

DESIGN FEATURES

- DISTRIBUTED DIGITAL MICROPROCESSORS
- AUTOMATED OPERATIONS
- CRT AND PLASMA DISPLAYS
- FIBER OPTICS
- SELF DIAGNOSTICS
- TRIPLICATED CONTROL SYSTEMS
- DATA HIGHWAYS
- EXPERT/AI SYSTEMS

REVIEW CRITERIA

- REGULATIONS
  - STANDARD REVIEW PLAN
  - RG 1.152
  - ANSI/IEEE 7-4.3.2 1982
- ADDITIONAL REVIEW GUIDANCE
  - PREVIOUS REVIEWS
  - IEEE STANDARDS 1012/729/730
  - IEC STANDARDS 880/987
  - MILITARY STANDARDS 2167
  - DESIGNER IN-HOUSE STANDARDS
  - FIPS/NSAC/FOREIGN

REVIEW ISSUES

- DIVERSITY
- ELECTROMAGNETIC INTERFERENCE
- EXPERT/AI SYSTEMS
- DESIGN CERTIFICATION
  LEVEL OF DETAIL
- PASSIVE PLANT CRITERIA
- PASSIVE PLANT HVAC
- COMMERCIAL DEDICATION
- SEGMENTATION
- SEPARATION/INDEPENDENCE
- DEFENSE IN DEPTH
- FAULT DETECTION/DIAGNOSTICS
- RELIABILITY

ONGOING DEVELOPMENT

- STANDARDS DEVELOPMENT
  - ANSI
  - IEEE
  - ISA
  - IEC
  - NRC - REGULATORY GUIDES AND SRP

- INTERNATIONAL TECHNICAL EXCHANGES
  - REGULATORY
  - VENDORS
  - UTILITY
  - RESEARCH
  - FRANCE / UNITED KINGDOM / CANADA / GERMANY / SWEDEN / NORWAY

- NRC RESEARCH
  - NRR USER NEEDS

ACRS SOFTWARE MEETING

SOFTWARE VERIFICATION & VALIDATION (V&V)

AGU R. ETS

SMARTWARE ASSOCIATES, INC

6 FEBRUARY 1991

file: V&V title

SOFTWARE V&V

- Ensure functional correctness
- Confidence that performs safety functions

CRITERIA FOR SAFETY SOFTWARE

- ANSI/IEEE Std 7-4.3.2 (1982)
- RG 1.152
- NUREG 0493
- OTHER CRITERIA

SUBJECT FOR AUDITS

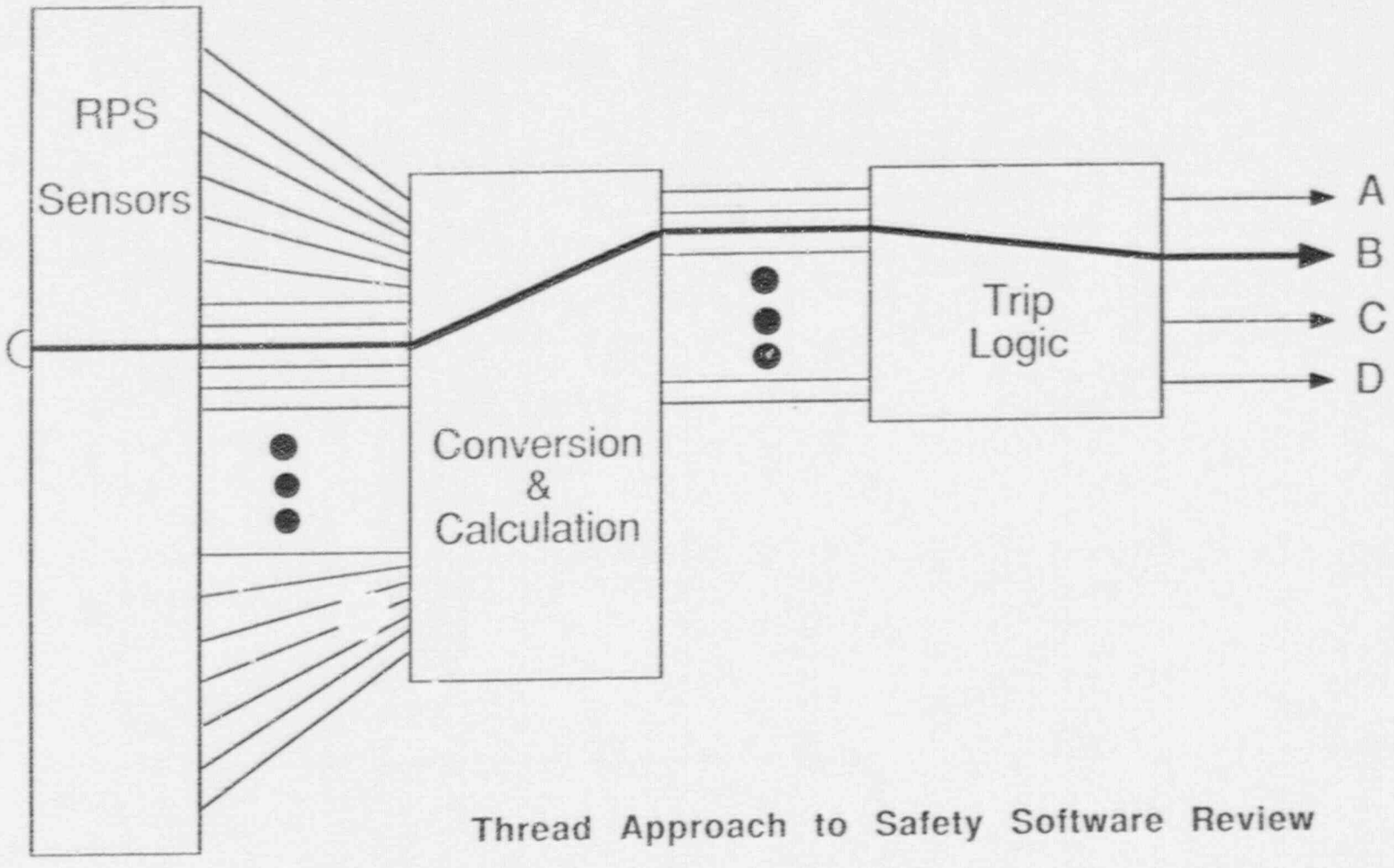
- Process for V&V
- Independence of V&V
- Application of V&V
- Requirements Documentation
- Configuration Management
- Development Methodology

file: Audit subjects

Audit Methodology

- Questions
- Thread Concept

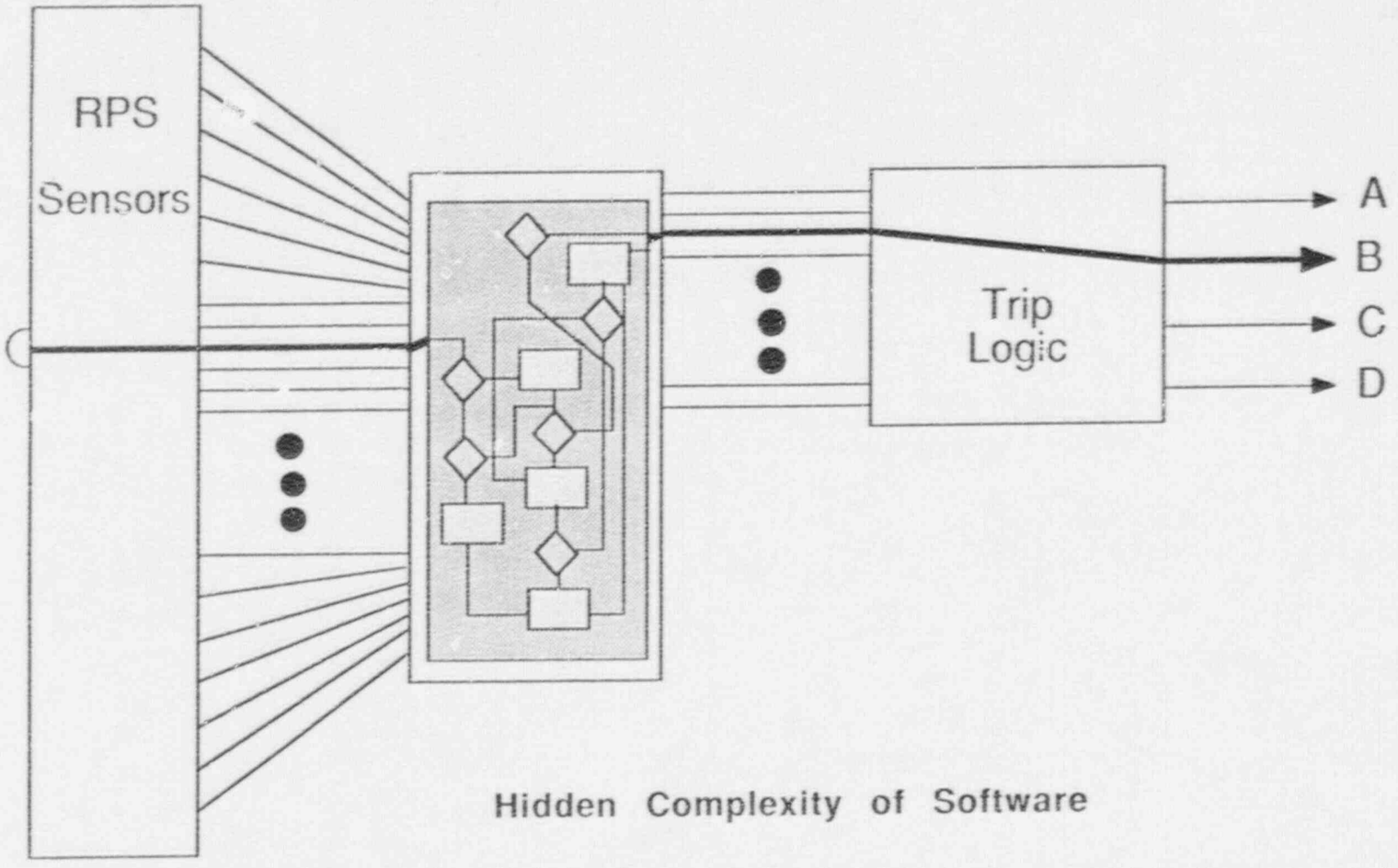
ACRS SOFTWARE MEETING



Thread Approach to Safety Software Review

file: Thread approach

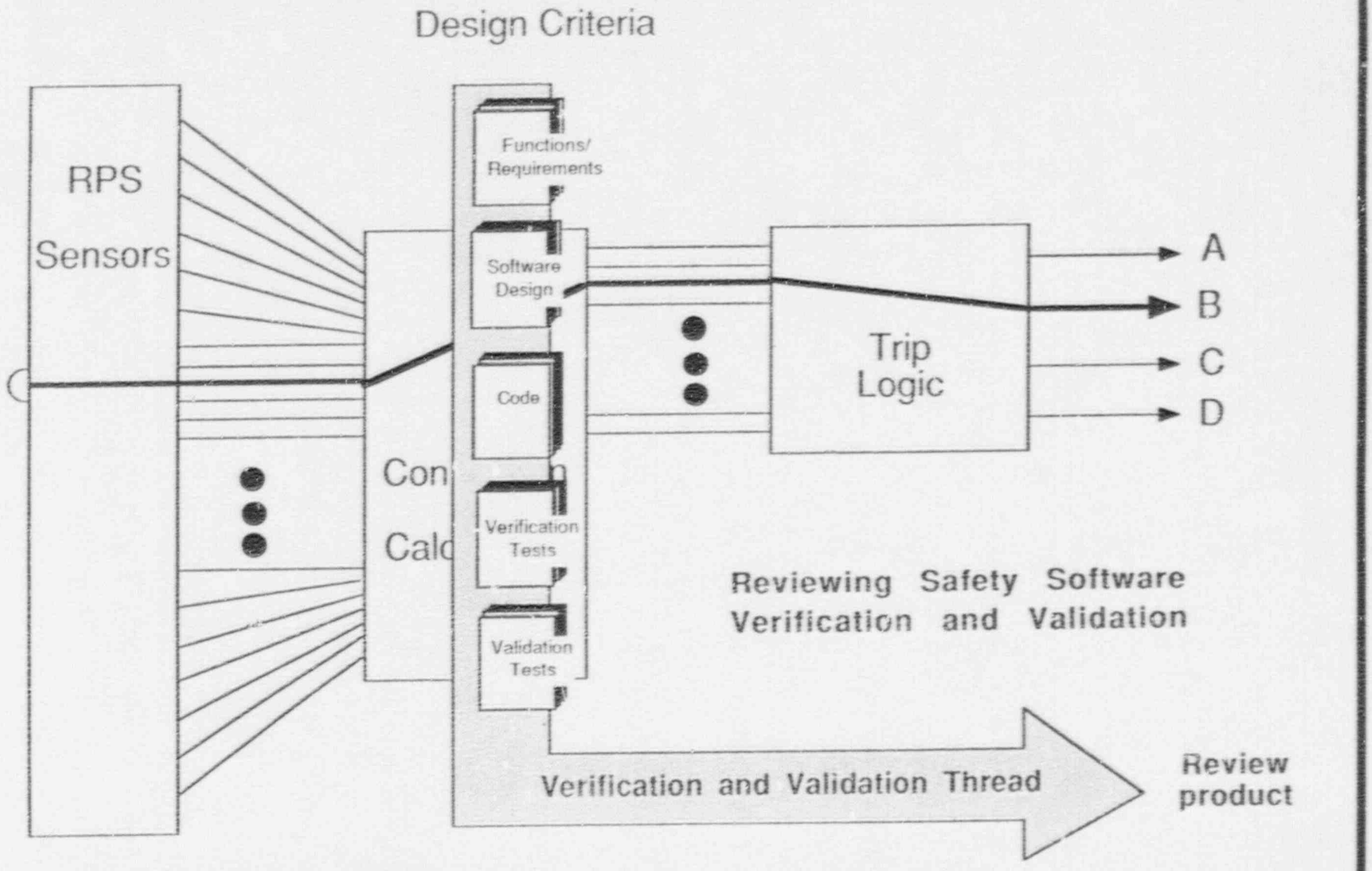
ACRS SOFTWARE MEETING



Hidden Complexity of Software

file: SW complexity

ACRS SOFTWARE MEETING



file: SW review

CONCLUSION

- V&V Is Proven Technique
- V&V Is Technology Independent
- V&V Can Be Supplemented With Other Standards

SOFTWARE VERIFICATION AND VALIDATION

(6)

VERIFICATION AND VALIDATION (V&V) IS A SYSTEMS ENGINEERING PROCESS EMPLOYING A RIGOROUS METHODOLOGY FOR EVALUATING THE CORRECTNESS AND QUALITY OF THE SOFTWARE PRODUCT THROUGH THE SOFTWARE LIFE CYCLE.

VERIFICATION

THE COMPARISON OF THE STAGE-BY-STAGE SOFTWARE DEVELOPMENT TO DETERMINE THAT THERE IS A FAITHFUL TRANSITION OF ONE STAGE (SUCH AS THE DESIGN) INTO THE NEXT STAGE (SUCH AS THE IMPLEMENTATION)

VALIDATION

COMPARES THE SOFTWARE REQUIREMENTS SPECIFICATIONS WITH THE FUNCTIONS IMPLEMENTED BY THE COMPUTER PROGRAM IN THE COMPUTER HARDWARE. ALSO PROVIDES ASSURANCE THAT THE OVERALL ACCUMULATION OF THE UNDESIRED STAGE-TO-STAGE SIDE EFFECTS HAVE BEEN CORRECTED.

(7)

FRENCH N4 I&C SYSTEM

FOR

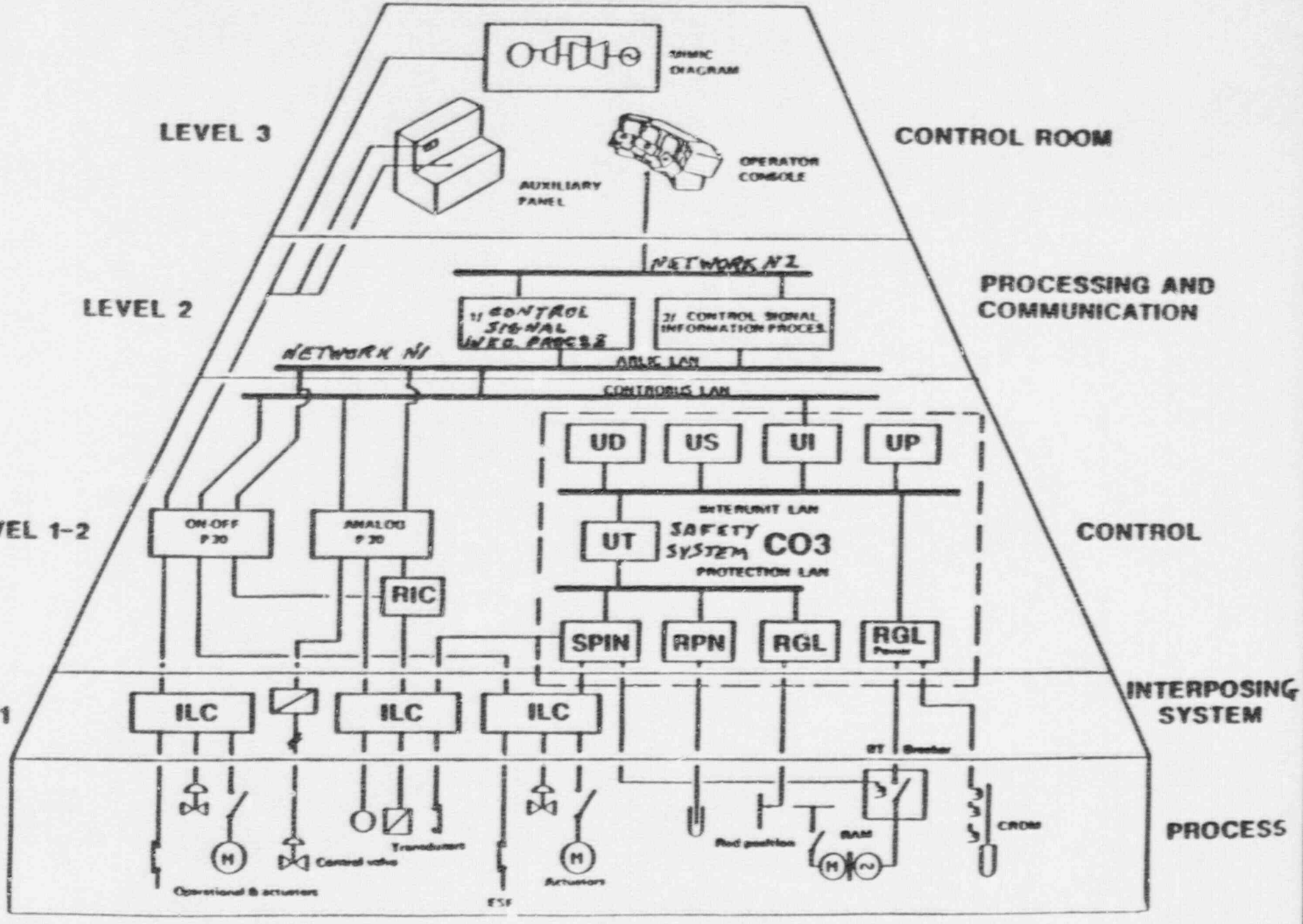
CHOOZ-B1 NPP

SUSPECTED PROBLEM

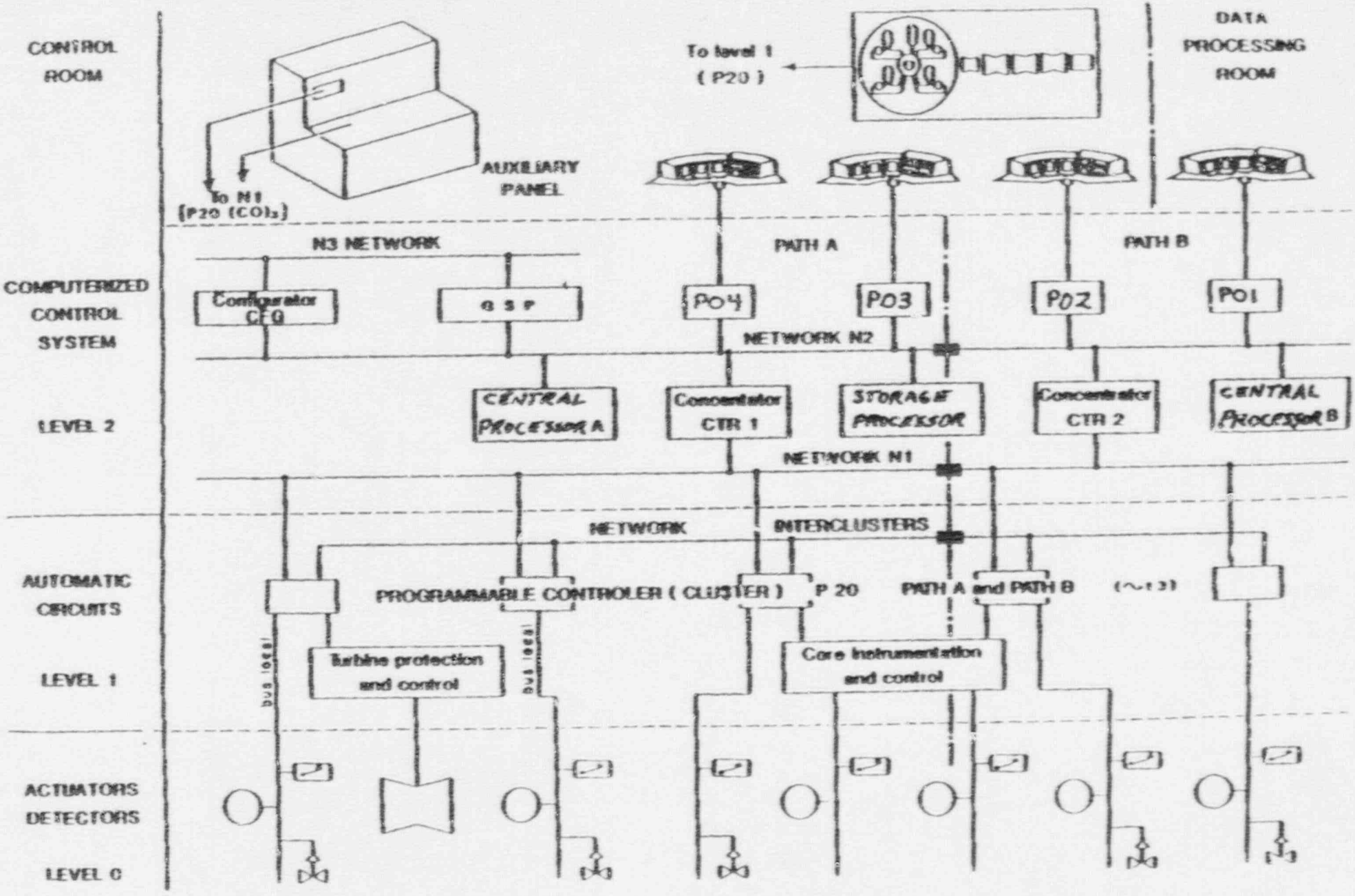
J. GALLAGHER SICB

2/6/91

N4 NPP I&C ORGANISATION



N4 INSTRUMENTATION AND CONTROL SYSTEM



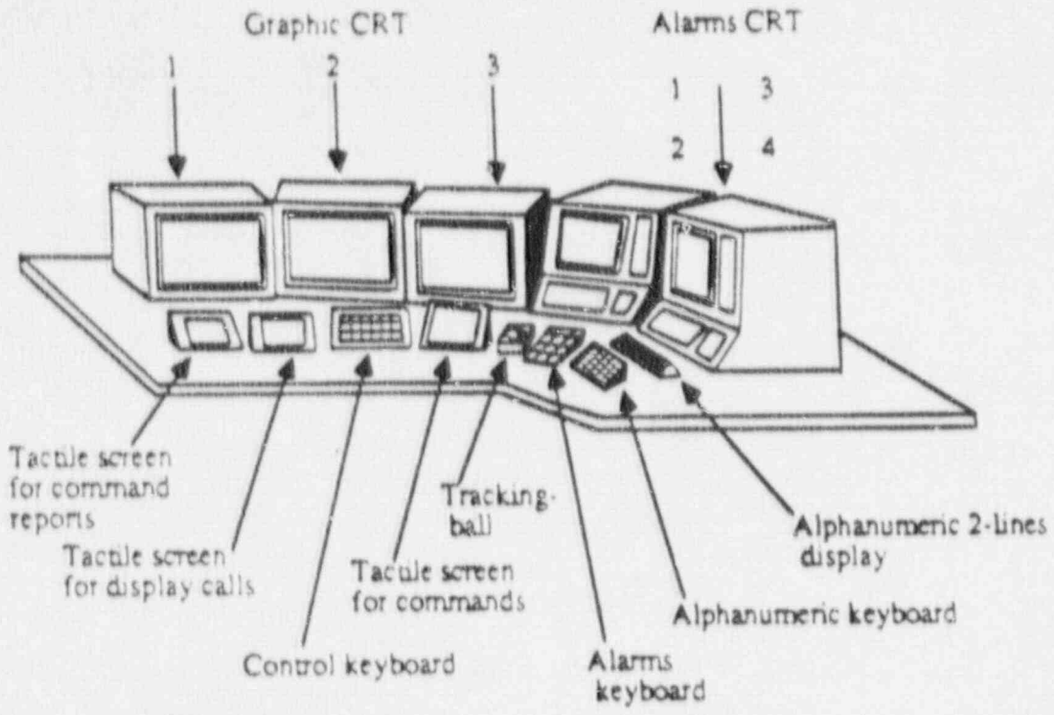


Figure 88 - CHOOZ B1 NPP CONTROL ROOM

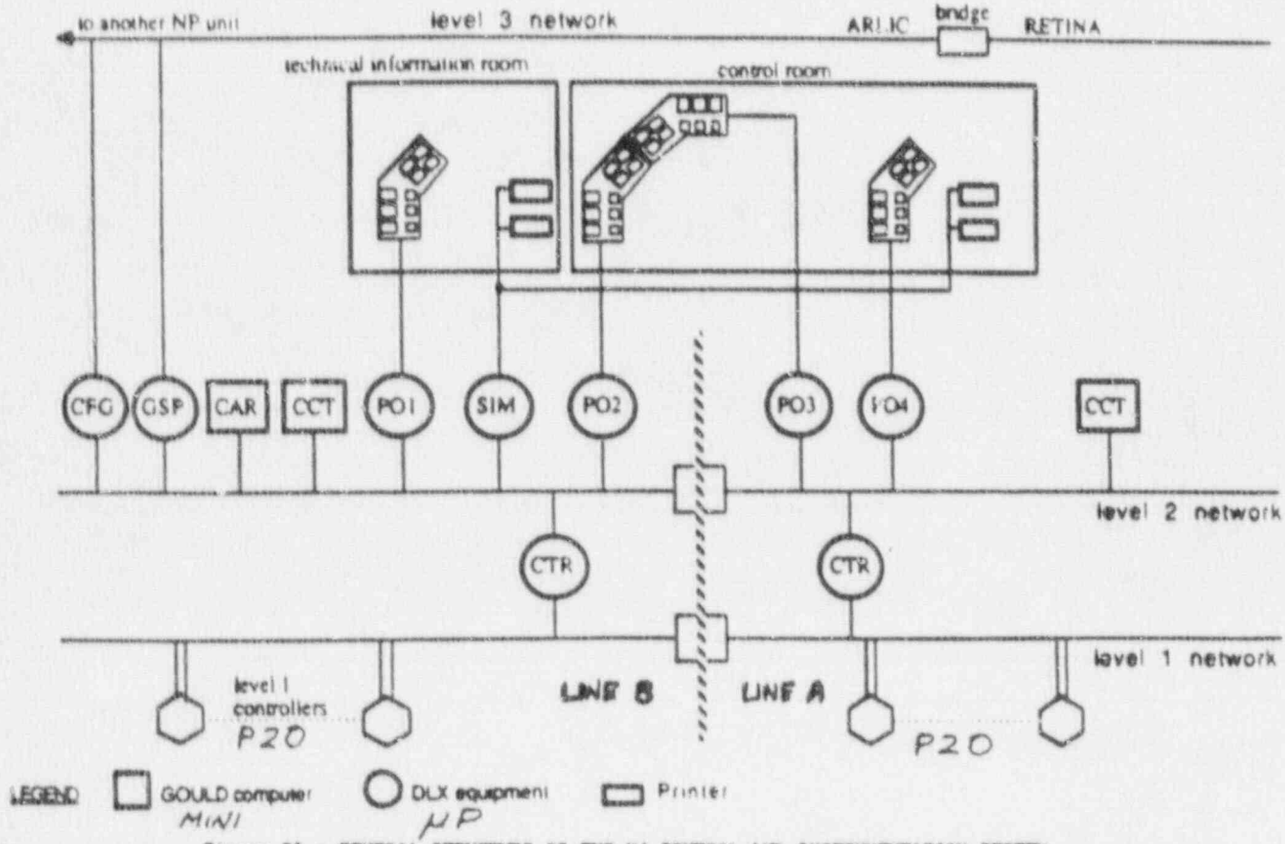
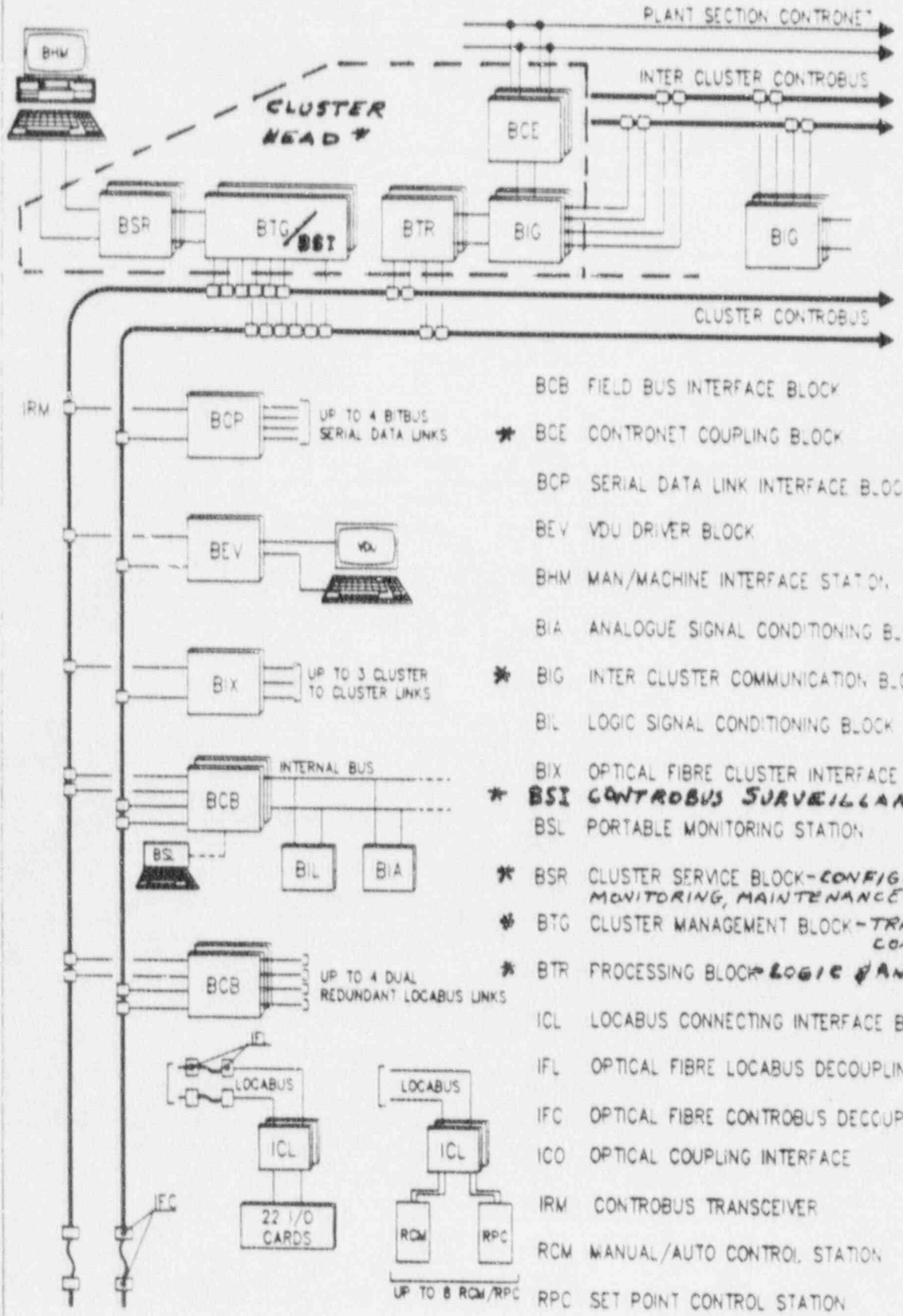


Figure 89 - GENERAL STRUCTURE OF THE N4 CONTROL AND INSTRUMENTATION SYSTEM

P20 CONTROBLOC

TYPICAL CLUSTER ORGANISATION WITH MNEMONICS



BCB FIELD BUS INTERFACE BLOCK

* BCE CONTRONET COUPLING BLOCK

BCP SERIAL DATA LINK INTERFACE BLOCK

BEV VDU DRIVER BLOCK

BHM MAN/MACHINE INTERFACE STATION

BIA ANALOGUE SIGNAL CONDITIONING BLOCK

* BIG INTER CLUSTER COMMUNICATION BLOCK

BIL LOGIC SIGNAL CONDITIONING BLOCK

BIX OPTICAL FIBRE CLUSTER INTERFACE BLOCK

* BSI CONTROBUS SURVEILLANCE

BSL PORTABLE MONITORING STATION

* BSR CLUSTER SERVICE BLOCK - CONFIGURATION MONITORING, MAINTENANCE

* BTG CLUSTER MANAGEMENT BLOCK - TRAFFIC CONTROL

* BTR PROCESSING BLOCK - LOGIC & ANALOG

ICL LOCABUS CONNECTING INTERFACE BLOCK

IFL OPTICAL FIBRE LOCABUS DECOUPLING BLOCK

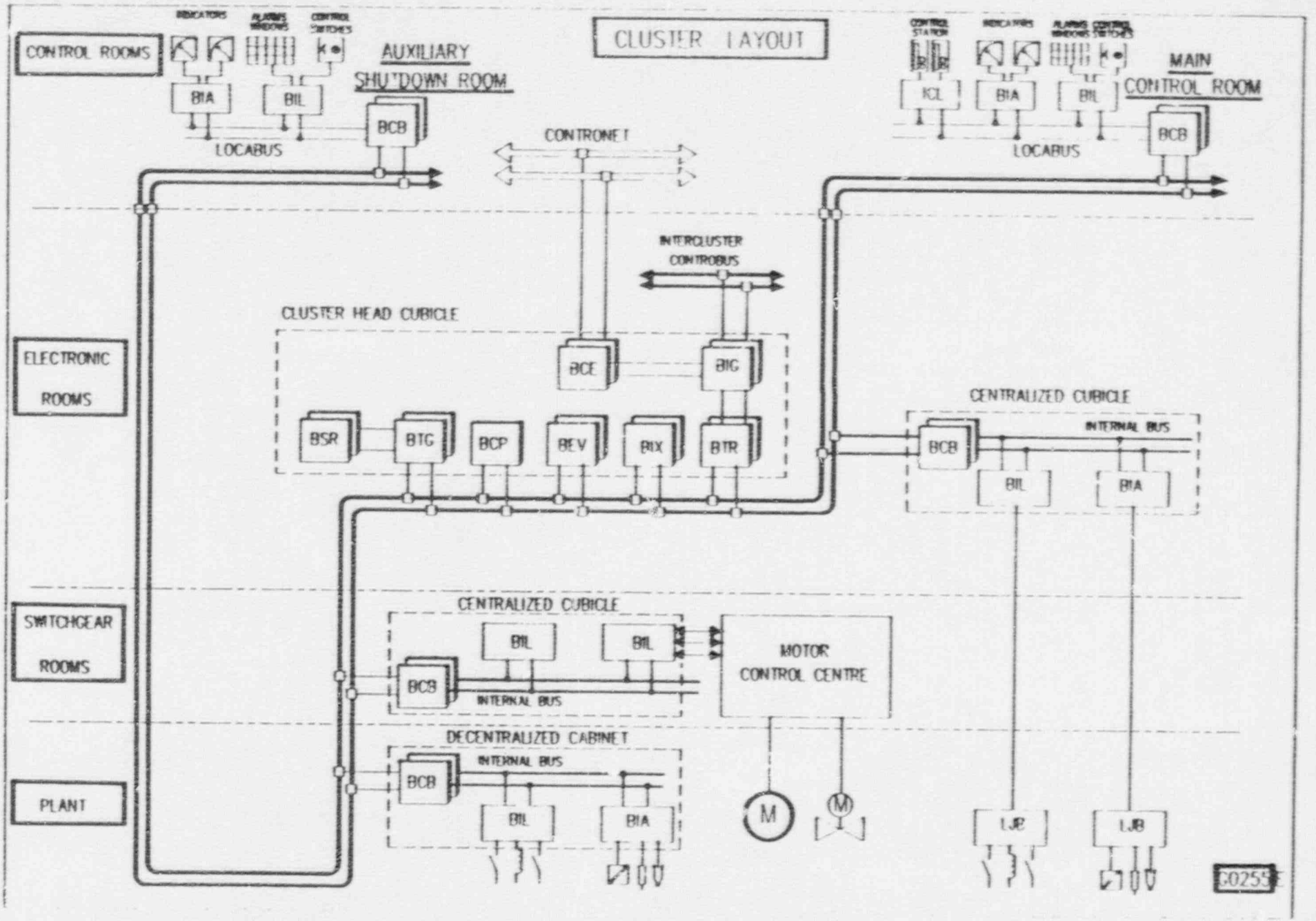
IFC OPTICAL FIBRE CONTROBUS DECOUPLING BLOCK

ICO OPTICAL COUPLING INTERFACE

IRM CONTROBUS TRANSCEIVER

RCM MANUAL/AUTO CONTROL STATION

RPC SET POINT CONTROL STATION


CLUSTER LAYOUT
CONTROL ROOMS
INDICATORS
ALARMS WINDOWS
CONTROL SWITCHES
AUXILIARY SHUTDOWN ROOM
BIA
BIL
BCB
LOCABUS
CONTRONET
CONTROL STATION
ICL
MAIN CONTROL ROOM
INTERCLUSTER CONTROBUS
CLUSTER HEAD CUBICLE
BCE
BIG
BSR
BTG
BCP
BEV
BIX
BTR
CENTRALIZED CUBICLE
INTERNAL BUS
ELECTRONIC ROOMS
SWITCHGEAR ROOMS
MOTOR CONTROL CENTRE
DECENTRALIZED CABINET
PLANT
M
LJB
C0255

N4 - CAPACITY AND PERFORMANCE OF SURVEILLANCE & CONTROL SYSTEMS

SYSTEM MONITORS & CONTROLS 10,200 ITEMS

DIGITAL CONTROLS - 8400
(ON-OFF)

ANALOG CONTROLS - 330
(CONTINUOUS)

DIGITAL DATA

55000 @ 1 SEC
10500 @ 0.05 SEC
DIGITAL RATE = 265,000/SEC

ANALOG DATA

100 @ 0.5 SEC
900 @ 2 SEC
1450 @ 20 SEC
ANALOG RATE = 730/SEC

P20 SUSPECTED PROBLEM