ELICITING BACKGROUND INFORMATION FOR SAFETY-CRITICAL SOFTWARE SPECIFICIATION

John C. Knight
Luís G. Nakano
Ambar Sarkar

Computer Science Report CS-95-42
September 1995

ELICITING BACKGROUND INFORMATION FOR SAFETY-CRITICAL SOFTWARE SPECIFICATION

John C. Knight, Luís G. Nakano, and Ambar Sarkar

(knight | nakano | ambar)@virginia.EDU

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

Contact Author:

John C. Knight
Department of Computer Science
University of Virginia
Thornton Hall
Charlottesville, VA 22903

(804) 982-2216
knight@Virginia.EDU

Keywords: Requirements specification, safety-critical systems, software specification.

ABSTRACT

Experience has shown that many defects in software systems are introduced during the specification phase. For safety-critical systems this is significant since it indicates where progress might be made in improving dependability. In this paper we argue that one of the reasons that defects are introduced during specification is that the specifier does not have an adequate understanding of the background and context of the system being specified. In order to provide a better basis for specification, we introduce the notion of *prespecification*—the determination and documentation of as much of the relevant background and context information as possible for the benefit of the specifier.

The specifier is often a software engineer. Though qualified to build software, he or she is unlikely to have a detailed knowledge of the application domain. Quite inadvertently, software engineers are likely to overlook aspects of the system that may have direct or indirect safety consequences. Application experts, on the other hand, are typically not trained to build software and therefore to understand its safety implications. To develop correct specifications, it is essential that there be precise communication between the domain expert and the software engineer.

In this paper, we propose the introduction of a phase in the software lifecycle for safety-critical systems that should precede the requirements analysis and documentation phases. We call this phase the *prespecification phase*. In this phase, pertinent background information regarding the software to be developed is collected and documented without undertaking a full-fledged requirements elicitation. The document that is generated is called the *prespecification* document.

We have built a prespecification document for a realistic safety-critical application. This application is a digital control system for the University of Virginia's Research Nuclear Reactor and a software specification for the digital control system is now being developed using the prespecification. We use the this prespecification as an example throughout this paper.

INTRODUCTION

Experience has shown that many defects in software systems are introduced during the specification phase. In some cases, measurement has revealed that the majority of defects are introduced during specification [1]. For safety-critical systems this is significant since it indicates where progress might be made in improving dependability.

In this paper we argue that one (but perhaps one of many) of the reasons that defects are introduced during specification is that the specifier does not have an adequate understanding of the background and context of the software being specified. In order to provide a better basis for specification, we introduce the notion of *prespecification*—a lifecycle phase that follows planning and precedes requirements analysis and specification in which as much of the relevant background and context information as possible is gathered and documented for the benefit of the specifier. The prespecification document is organized and structured to maximize its value as a reference source for the specifier.

Safety-critical applications are complex and often unfamiliar to the specifier, and it is quite possible that errors are introduced because the specifier is unaware of significant concepts from, common definitions in, or obscure but important detail about the application domain. This argument is given some credibility in part by the very subtle and unexpected nature of the causes of most accidents and in part by the generally accepted notion that many of the problems in specification are problems of communication.

The specifier is often a software engineer. Though qualified to build software, he or she is unlikely to have a detailed knowledge of the application domain. Quite inadvertently, software engineers are likely to overlook aspects of the system that may have direct or indirect safety consequences. Application experts, on the other hand, are typically not trained to build software and therefore to understand its safety implications. To develop correct specifications, it is therefore essential that there be precise communication between the domain expert and the software engineer.

The specification of a software system's requirements is often regarded as the vehicle for this communication. A specification contains various types of information but this information can be broadly broken down into two categories:

- functional and non-functional requirements and,
- essential *background information.*

The background information contains definitions and explanations needed to understand the remainder of the specification. Traditionally, the specifier focuses on eliciting the functionality requirements, and obtains the background information on a "need-to-know" basis. Thus both actual requirements and background information are obtained in a single phase.

As elements of two case studies†, we have worked on specifications for two non-trivial

†. The first case study is a robotic neurosurgical device, the *Magnetic Stereotaxis System* (MSS) [2]. The second case study is focused on the *University of Virginia's Research Nuclear Reactor* (UVAR), where we are engaged in the specification of a digital control system.

applications. As a result, we have concluded that gathering both requirements and background information in one phase is unsatisfactory, especially when the background information is given a secondary role. We consider background information to be important enough to be obtained in a systematic manner, instead of the ad hoc manner that is typically employed.

We claim that a separate activity is needed in which, to the extent possible, the necessary background information is collected. While this information might not be complete, this activity will force the specifier to understand and document relevant and available information in an organized manner thus avoiding many potential mistakes that can occur if such background information is overlooked. The resulting document helps to bridge the gap between the domain expert and the software engineer, and it provides in part the vehicle for the precise communication that is required between them. We are not aware of any approach, especially in the domain of safety-critical systems, that stresses the relative importance of background information.

There is an activity that is quite common in hardware engineering in which the interface specification for some element of a system is defined and documented. This activity has some similarity to the prespecification concept that we are advocating but lacks the interrelationship between the various functions that the interface provides, the theory behind the various functions, and the possible role of human operators or users.

We have built a prespecification document for one of our case studies, the University of Virginia's Research Nuclear Reactor (UVAR), and a specification for a digital control system is now being developed using the prespecification. We use the UVAR prespecification as an example throughout this paper.

In the next section of this paper, we discuss our rationale for prespecification. We then present more details of the prespecification concept. Next we discuss a case study of prespecification, and finally we present our conclusions.

RATIONALE FOR PRESPECIFICATION

By the term *background information*, we mean the information that is required by someone who is not an expert in the domain to understand the application and its associated environment. Such information does not include anything that would be considered normally a requirement or part thereof.

The prespecification should explain sufficient about the background and context of the software to be developed that, under ideal circumstances, the specifier is able to get all his or her questions answered about the application context from this document. The prespecification establishes the breadth and depth of the understanding that is needed by the specifier.

As an example of the type of information to which we refer, consider the extensive sensing instrumentation used by a typical control system. Details of each sensor such as its purpose, its range, its claimed accuracy, its resolution, its data representation, its timing constraints, its failure modes, its use within the system, its level and type of redundancy, and the means by which its value is displayed for the operator (if the value is displayed)

are examples of the kind of information that should be documented systematically for reference during specification.

An approach that one might consider is to depend upon application experts being involved in any software development project and merely asking for background information as necessary. The information could be documented if necessary as it was acquired. We suspect that this is what typically happens in practice yet the high level of defects in specifications remain and so we infer that it is not a perfect solution. It also suffers from the deficiency that it leaves the acquisition of vital background information to chance.

A systematic process for acquiring and organizing background information is essential. However, in most current approaches to software specification, background information (a) depends on reference information that tends to be scattered through a variety of documents, (b) is obtained in a rather ad hoc manner during lifecycle phases of which it is not the focus, and (c) is not identified as a separate item. We consider this to be quite unsatisfactory for large-scale, safety-critical applications where omissions and misunderstandings can be serious. We expand on our rationale in the remainder of this section.

Scattered Information

It might be argued that a prespecification phase is not needed since the requisite information is available in existing documents. Typically, however, background information is scattered widely in these existing documents, and, although the documents probably (though not necessarily) contain the relevant information, for any realistic safety-critical application they are inadequate for the following reasons:

- Existing documents tend to be organized for the convenience of someone other than the specifier. For example, a document describing system peripherals might be organized with an emphasis on the physical organization of the electrical wiring although buried in the document are details of the input/output hardware ports being used.
- Existing documents are written assuming a technical background that is different from the specifier's. For example, a description of an application might be written assuming that it will be read by an application engineer rather than a software specifier. Essential definitions are likely to be omitted, therefore, because they are assumed known, and a level of sophistication in areas such as continuous mathematics might be assumed because it is common in the application domain.
- Existing documents often overwhelm the software specifier with information.

Using existing documentation, a great deal of time is often spent tracking down the relevant sources of information, and, when found, the information is frequently in a form that does not readily satisfy the needs of the specifier.

We infer from the inadequacies of existing documentation that (a) a comprehensive source of background information is required and that (b) this background information has to be organized and written keeping in mind the typical software engineer's knowledge and experience (or lack thereof) in the safety-critical application domain. Having such

information in one place opens it to the scrutiny of both the software engineer and the domain expert. Once consensus is reached on the content, the prespecification can act as the authoritative repository of system-related knowledge—an asset that is likely to be very valuable to both parties. The domain expert will be satisfied since the necessary information that is considered important for the safe operation of the system will be in a document that is both understandable and possibly amenable to analysis by the domain expert. The software engineer will be satisfied since the related information that one would need to develop the software will be in one place.

Focus On Requirements

How then should such a document be developed? We contend that it should not be during requirements analysis and specification. Requirements analysis and specification are typically the first phases of the software lifecycle and they focus on determining and documenting the software requirements of a system. Some background information is gathered systematically because it is related directly to functionality in an obvious way. In structured analysis, for example, file formats have to be defined and data dictionaries developed to support the dataflow models. Even in a case like this, however, important background information can be missed. For example, the expected size of a file, the expected variability in size, its access pattern, and so on are not usually defined as part of a structured analysis yet they can affect the details of a system's software specification.

In safety-critical systems, safety concerns often arise from sources that are orthogonal to the functional requirements and therefore cannot be deduced from the specifications of the functionality. For example, a term commonly used in the application domain that is used in software engineering *but with a different meaning* is a recipe for disaster.

The specifier should know as much relevant information about the system as is practical before he or she attempts to develop the specification. This would avoid many errors in the specification from the very beginning thereby reducing the cost of building a specification. Since the specifier would be aware of the background, the requirements analysis and specification phases would be more focused and efficient. The specifier would be asking relevant and "pertinent" questions sooner instead of prolonging the requirements analysis phase.

We note that the safety of a system may be affected by parameters that are often not a part of the system's software requirements. This is where the relative emphasis on background rather than the requirements is important. Not being constrained by having to record only software requirements, the software engineer is free to obtain information that would have otherwise been dismissed as not being very relevant to the functionality of the system.

Finally, we observe that specification detail often rests on assumptions associated with background information. By devoting time to the prespecification, there is a greater chance that these assumptions will be clearer from the very beginning rather than becoming clear when it may be difficult to effect a change because activities are advanced in the specification development cycle.

A Separate Lifecycle Phase

The domain expert and the specifier need to establish a common ground of understanding and this should be systematized rather than leaving it to ad hoc and need-to-know approaches. This ground should be created by the specifier interacting with the domain expert to acquire the relevant information. This information should be then organized and presented back in a manner that the domain expert can easily understand and inspect. Once this process has been iterated to the satisfaction of both parties, it is established as a definitive, living document. Any future references to the system should be obtained from this document, and if in future this document does not suffice, careful updates should be made to it and to the decisions that were made based on it. A major advantage to this approach is that the information relating to assumptions that were made regarding a system will be in one place, making the process of determining the effect of the update more feasible.

Existing Specification Techniques

There are certain aspects of existing specification techniques that are designed to develop information relevant to a specification and which have similarities to what we call prespecification. Though very useful and essential within the specification context of their definition, they are not prespecification as we define it and so do not eliminate the need for prespecification.

An example is the introduction of systematic glossary information and systematic input/output port definitions in the A7 specification technique [4]. The role of the glossary in an A7-style specification is to centralize important definitions so that the meanings of important terms will be clear and shared by the various users of the specification. The role of the port definitions is to define port information in a consistent and thorough manner.

These techniques are major contributions to specification quality but they do not replace the need for prespecification. In the case of the glossary, whether to include information is determined to a large extent by the specifier rather than the application expert although the application expert will obviously be involved in determining the information itself. In addition, glossaries fail to show what can only be described as the "big picture" and the system-level areas of safety concern.

A second type of information that has some similarity to prespecification is the result of a hazard analysis [5]. The goal of a hazard analysis is to identify system states that, if entered, could lead to an accident. Thus the role of hazard analysis in understanding the safety elements of a system is clear. But again, we note that a hazard analysis, at best, only adds to the specifier's understanding of the application domain and so this understanding is very likely to remain incomplete. However, it is this understanding that will enable the specifier to make appropriate choices, ask the right questions, and know when vital information is missing during specification itself.

PRESPECIFICATION CONCEPT

In this section, we examine the concept of prespecification in more detail. The ideas

expressed are the result of our experience with the prespecification we have written and its use so far in our case study. The ideas are, therefore, preliminary and heavily influenced by the application domain in which we are working.

We begin by enumerating the properties that we have found to be important. We then review the structure and content that we have developed for a prespecification document, and finally review some aspects of how the prespecification might be used.

Properties

A prespecification has to serve as a major communications channel between engineers in different disciplines. To do this effectively, the prespecification has to be prepared carefully and possess the following essential properties:

- The prespecification should be as complete as possible in the sense that everything about the background and context of the system to be built should be included.
- A combination of formal and informal notations should be used such that the prespecification is understandable and amenable to analysis as a whole by all the engineers involved (application, software, etc.) and so that the engineers conform in their views of the system. Tables are a simple yet very helpful formalism.
- The prespecification should be as precise as is deemed necessary by the engineers involved. Precision is valuable but the existence of a topic in the prespecification is the source of the major benefit. By being alerted to the topic, the specifier is aware that seeking further precision might be necessary.
- The level of detail should be set so as to satisfy the various parties involved. However, we note that if something is not obviously relevant but any party thinks it might be, it should be included if resources permit. At the very least, references to other documents should be included at appropriate points.
- Detail that is required but which is for some reason unavailable can be omitted as long as it is clear that detail is missing. Detail might be unavailable because it is unknown or to-be-determined. Not knowing the detail is acceptable. Not knowing that it is not known is not.
- Detail that is required but that is unclear or known to be erroneous should be included and marked as possibly wrong.

Structure And Content

The structure we have evolved for the prespecification is quite simple but was derived from our experience with one case study. It is likely that additional experience will refine the structure.

Our structure focuses mainly on identifying systematically the equipment with which the computing system will be involved. First, we include a section that describes the "big picture"—what is this system and what is it for. Next we include sections that document sensors, actuators, and input and output devices for operator information and control. In

order to try to understand how the system will function, we include a section that documents how the system will interact with the outside world including human operators where appropriate.

There is obviously a need to include sections within a prespecification that document topics that are application-specific. For example, in our case study of the research nuclear reactor we found it necessary to include a section in the prespecification that documents the security system. This system is partly automatic and partly manual but understanding it was critical to understanding why certain other parts of the system have to operate in certain ways.

We include within our prespecification structure several sections that are designed to function as repositories so that detail will not be lost. For example, we include sections to hold pending technical questions, apparent or real errors in documents provided by the domain experts, and a glossary of terms that grows as terms are defined.

Development

Since the basic purpose of a prespecification is to document background information in the application domain, one could reasonably ask why the specifier could not learn the domain the way domain experts do, perhaps by taking some type of course. The problems with this approach are many: there might not be a course available; a course might not cover the necessary material; or a course might cover too much material.

The prespecification is just a collection of information and so the process of development is really just one of determining what information should be recorded and then acquiring it. The development process should start with a trivial requirements analysis phase so that the authors of the prespecification can become familiar with the overall system goals. For example, in the UVAR case study, we began by determining that the system to be specified was an enhanced version of the reactor's present electromechanical control system.

Once this level of requirements analysis is finished, the outline given in the previous section can act as a guide to determining how to proceed. In the final analysis, a great deal of effort has to be expended in reviewing existing documents and interrogating domain experts.

Use

Once finished, the prespecification plays a variety of roles. It can act as a reference source throughout the life of the associated software. But parts of the prespecification can become parts of other documents as development proceeds. For example, details of sensing equipment should be moved to the specification itself once work on the specification is underway.

A prespecification can play different roles in different types of system development. For example, when developing a new application, much of the background information will be on hand but still needs to be organized and explained for the benefit of the specifier. In the case of a replacement of an existing system, a prespecification can act as a focus for acquisition of scattered information. For software maintenance, a prespecifica-

tion can document information that would be hard to find in other ways because the original development engineers are no longer available.

Maintenance

A prespecification needs to be maintained but maintenance of a prespecification will be slightly different from the maintenance of other artifacts. Much of the content should be unchanging and require no attention except the correction of errors. Since the prespecification is documenting background information it is unlikely that very much of that background will change during software development. What is much more likely to occur is the realization that certain important background information was omitted during the initial development phase of the prespecification. Thus significant additions to the prespecification are likely over time.

A CASE STUDY IN PRESPECIFICATION

The ideas in this paper evolved from our case study of the University of Virginia's research nuclear reactor. As an element of our research in software engineering, we are developing the specification for an advanced control system for the reactor. By studying the issues that arise in a complex application such as this, we are endeavoring to better understand the issues involved in safety-critical systems.

A very important general point that has to be stated at the outset is that those involved in this case study had a reasonable lay knowledge of nuclear physics before the case study began. All were familiar with the general principles of fission reactors, their applications, and the extreme care that must be taken with their construction and operation. However, the complexity of the basic physics involved in understanding how a real reactor operates was a surprise to all involved.

The Application

The University of Virginia Reactor (UVAR) is a 2 MW thermal, concrete-walled pool reactor. It was originally constructed in 1959 as a 1 MW system, and it was upgraded to 2 MW in 1973. The reactor core consists of 20 to 25 plate-type fuel assemblies placed on a rectangular grid plate. The reactor uses water cooling and the core is immersed in a large-capacity (80,000 gallons) water pool. In a low power mode, cooling can be by natural convection. In a high power mode, cooling is by forced water flow. There are three scramable control rods for reactor shutdown, and one non-scramable regulating rod that can be put into an automatic mode. In this mode an analog control system uses the regulating rod to maintain reactor parameters.

A variety of sensors measure process variables including: 1) relative output by movable fission chamber; 2) neutron flux by ion chamber; 3) start-up neutron flux and period by BF_3 counter; 4) core inlet and outlet temperatures by thermocouples; 5) primary system flow by pressure gauge; 6) control and regulating rod positions by potentiometer; 7) gross gamma-ray dose by ion chamber; and 8) various levels (such as pool water height) by limit set switches.

Other than the regulating rod, operation of the reactor is by manual control. All indications of reactor operating conditions are presented on analog or digital meters, and/or on strip chart recorders. Operation of the reactor proceeds using detailed standard operating procedures (SOP's) with checkoff sheets to confirm that steps are completed with interlocks that prevent operation when certain steps are not done in proper order or equipment is not functioning.

Our goal with this case study is to develop the software specification† for a digital control system for the reactor that would provide the functionality of the current electromechanical/manual system plus some new capabilities. These new capabilities include more sophisticated displays, additional operational checks, and automation of certain functions that are presently manual.

Existing Documentation

As expected, there were a number of documents that provided much of the necessary background information. The existing documents that were reviewed in the development of the prespecification were the following:

- Standard Operating Procedures for the University of Virginia Reactor, 195 pages, no drawings.
- Emergency Plan Implementation Procedure, 90 pages (grouped as individual procedures), no drawings.
- Amendment 1 to the Revised Safety Analysis Report and Technical Specifications for Two Megawatt Operation - University of Virginia Reactor March 1971, 72 pages, 13 figures, 8 tables.
- University of Virginia Reactor Revised Safety Analysis Report, Revision 7 July 1989, 257 pages, 73 figures, 24 tables.

All of these documents were extremely well-structured and thorough but inevitably were written with a nuclear ~~engineer in~~ mind. They were also each written for a specific purpose as indicated by their titles and contained a great deal of material that was not immediately useful in developing the document that we sought. However, it was not difficult to understanding most of the related material which is somewhat surprising given our essentially non-nuclear engineering background.

When we tried to understand certain aspects of the reactor in detail, the information was complex enough that we had to refer to the documentation in addition to interrogating the domain experts. For example, information regarding the operator control panel instrumentation and layout was scattered in various documents. In some cases, display instrument documentation did not indicate directly what were the normal, abnormal, and impossible system operational ranges. This information is actually available indirectly and is common knowledge among the nuclear experts but was not apparent to us.

---

†. Note that we are only developing a specification. In keeping with NRC regulations, at no point will we have access to any reactor control equipment.

We spent a considerable amount of time trying to categorize the operator control panel instrumentation. We created labeled pictures of the control panel and integrated descriptions of the individual instruments so as to provide us with a systematic, structured source of details about the controls. This is discussed in more details below.

The Prespecification

The prespecification was developed over a period of about six months with contributions from a variety of people. The primary authors were two masters candidates, a doctoral candidate and a postdoctoral fellow in computer science at the University of Virginia. Technical information about the application was obtained from a variety of domain experts in nuclear engineering.

The prespecification is 64 pages long and the significant sections are:

- *Introduction*

  This section explains the goals of the document and summarizes the reactor system.

- *Control Panel*

  The entire control panel as presently constructed is described with details of each instrument and control.

- *Human Interaction*

  This section documents the role of the human operator in the present system, and the role of the researcher using the reactor for experimentation. Since control of the reactor is required to establish the necessary environment for an experiment, the researcher has to interact with the reactor operator. The activities of the operator and the researcher are described as procedural programs in an Ada-like pseudo code.

- *Security Concerns*

  This section enumerates the various security interlocks that are used in facility security.

- *System Details*

  The system details are broken down into systematic documentation of the system's sensors, actuators, interlocks, and alarms. In addition, the conditions under which the reactor must be shut down (the scram conditions) are elaborated.

- *Open Questions*

  This sections is a repository for as-yet unanswered questions. It provides a single location for storing this material so that these questions can be addressed systematically. Naturally, in the final form of a prespecification, this section should be empty.

- *References*

  This is a list of relevant reference documents.

- *Glossary*
  Definitions of important terms.
- *Possible Mistakes in Original Documentation*
  This section is a repository for comments by the developers of the prespecification about what they consider to be errors in the reference documents.
- *Figures*
  This section includes copies of figures that we considered significant that were contained in the existing documentation.

Notations

We found that for precise representation and presentation of the information we gathered, very simple constructs sufficed. The main concern was that we are able to note down pertinent information in the most suitable, clear, and readable, i.e., communicable form. While we did not use formal notations specifically, we were careful to record our information unambiguously using simple structural constructs and clear English phrases. If we had felt a need for more formal notations, we would have used them.

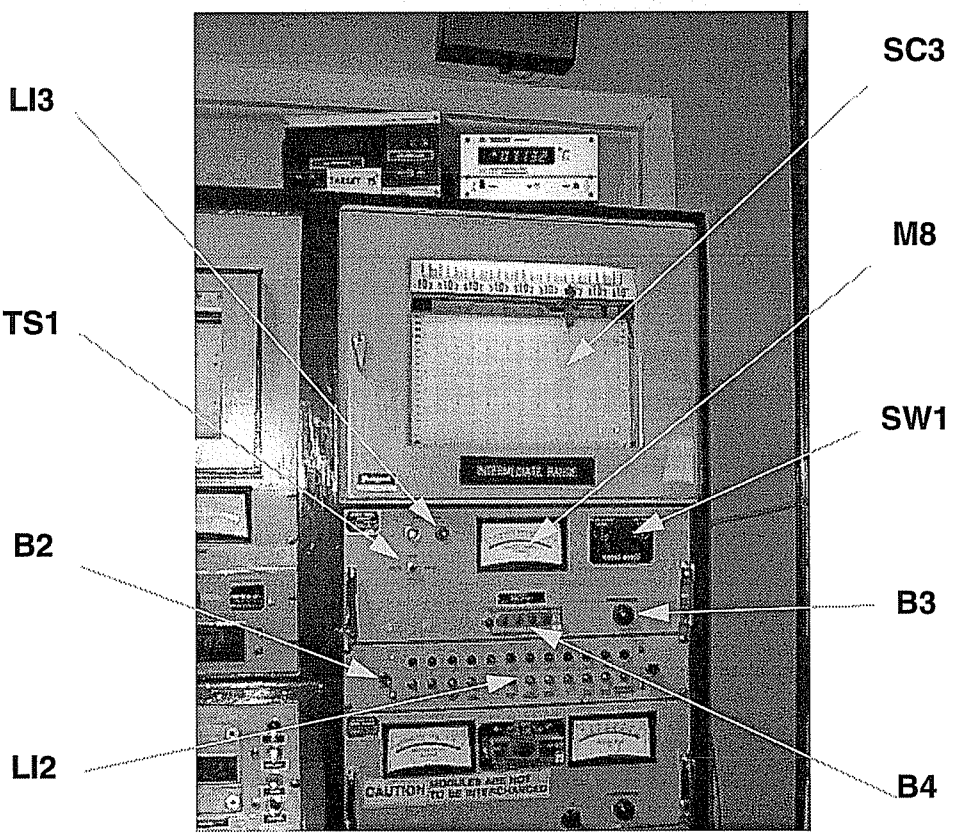

Fig. 1. Photograph of upper right section of operator's console

Instrument ID:	SC3.
Instrument Type:	Paper strip chart recorder.
Information Displayed:	Intermediate range (see glossary).
Units/Scale:	Counts per second/7 decade log scale(10^3 to 10^{10}).
Purpose:	Show trend for intermediate range drawer output.
Description:	Paper velocity: 6" per hour Valid for normal operation
User Function:	Output

Fig. 2. Intermediate range strip chart recorder description

The vast majority of the material in the prespecification is in the form of either tables or bulleted lists. Very little natural language text is used. To document the operator's console and its instrumentation, a variety of photographs were taken, labelled, and entered into the document. One photograph shows the entire control panel to establish the layout. The panel is then broken down into six subsections and a photograph of each subsection is included. These later photographs are sufficiently detailed that instrument labels can be read. The labels that we have superimposed onto the photographs provide the link between the instrument and the table describing the instrument. Fig. 1 is an example photograph showing what is referred to as the Upper Right section of the operator's console.

Precision of notations helped us organize our thoughts and ask intelligent questions, and the simplicity allowed us to phrase our questions in a manner that was understandable by people with a diverse background. We did not see any merit in formalizing certain aspects of the system at this phase, since our understanding of them was not yet comprehensive.

An example of the value of limited precision came from the reactor operator interaction algorithm described in Ada-like pseudo code. A simple inspection of the description of the researcher interaction enabled us to realize that:

- the software engineer had misinterpreted the interaction, and
- there existed the possibility of multiple access to the facility that had not been considered earlier.

A basic tabular structure was chosen for the description of instrumentation. It seemed to be an optimal combination of both understandability and precision. Each row represented an attribute of the instrument and the attributes chosen were mostly suggested by the prespecification developer. The strip chart recorder identified in the Fig. 1 as SC3 is documented using the table shown in Fig. 2.

Forcing us to write the description in the tabular format made us discover several ambiguities in what we had written down from our initial discussions. Once we had reor-

ganized our descriptions, our questions became more specific and therefore the answers we received were more precise.

Verification

Verification of our prespecification document was by inspection. During development, each section was reviewed for accuracy by the authors and questions were answered by reactor personnel as they arose.

After the document was thought to be complete by its authors, it was subjected to a thorough inspection by a staff member of the reactor facility (a senior reactor operator). The result of that inspection is the main result that we report:

> *The inspection revealed that the prespecification as originally written contained a large number of significant errors.*

A wide variety of different types of error was found. There were errors of omission in which, for example, certain instruments were not documented; errors of detail in which, for example, parameters were documented with the wrong values; errors of fact in which, for example, information was missing; errors of phrasing in which, for example, equipment was described very poorly; and errors of status in which, for example, equipment was documented that no longer exists.

Some, perhaps many, of these errors are attributable to less than perfect work by the authors and, upon reflection, some errors are quite obvious. Other errors no doubt could have been found (and should have been found) by a more systematic, rigorous inspection by the authors. But many of the errors were subtle misunderstandings of a complex system that could only be found by an experienced domain expert. It is our opinion that this is a clear indication of the importance of prespecification.

It is important to keep in mind the purpose of the document in which these error were found. This was not a specification, it was merely a prespecification that documented the necessary background information. The reasons that we consider this result to be the most important are twofold. First, even with the relatively narrow goal of describing background information, significant errors were made. Had we been trying to build a specification at the same time, it is quite likely that we would have neither understood the background nor been aware of the defects that would have crept inevitably into the specification as a result.

The second reason why we consider this result to be our most important from this project is that we are now in a position to correct the errors. Once this is finished, we will have a comprehensive reference document with which to proceed with specification. Our hypothesis is that this will help reduce specification errors considerably.

A second significant result that we report is that the very process of background capture led to a much more thorough understanding of the background. Researchers and practitioners in formal specification have reported that the systematic thought process that is imposed by the construction of a formal specification tends to improve the quality of a specification. This is in addition to the benefits brought about by a formal notation. We

observe the same effect in background capture. By focusing on understanding the context, a thought process was evoked that caused the authors to search systematically for the information needed.

Finally, we note that the goal of capturing only background information was known to be important in this project but proved hard to address. Throughout the development of the prespecification, there was a constant tendency to be concerned with specification information.

CONCLUSIONS

We conclude that background and context information is vital to the process of software specification, especially for safety-critical systems. We refer to this information as prespecification. We also conclude that the acquisition of this information is best undertaken by a lifecycle phase that precedes requirements analysis. Our justification for these conclusions is experience gained in a case study in which we have developed a prespecification for a realistic safety-critical application.

We are aware that the concept of prespecification has not been explored completely by the research reported here. Additional research is need to determine, for example, the best way in which to acquire and organize the associated information. In addition, it will be necessary to determine the effect, if any, of prespecification on specification quality.

ACKNOWLEDGMENTS

It is a pleasure to thank the following colleagues at the University of Virginia for their assistance with the work described in this paper: M Elder and G. Ferrer of the Computer Science Department, R. Uddin of the Mechanical, Aerospace, and Nuclear Engineering Department, and C. Bly, J. Carroll, B. Hosticka, D. Krause, and R. Mulder of the University of Virginia's research reactor facility. It is also a pleasure to acknowledge many helpful suggestions from L. Beltracchi and R. Brill of the U.S. Nuclear Regulatory Commission.

This work was supported in part by the U.S. Nuclear Regulatory Commission under grant number NRC-04-94-093, by the National Science Foundation under grant number CCR-9213427, and in part by NASA under grant number NAG1-1123-FDP. This work was performed under the auspices of the U.S. Nuclear Regulatory Commission. The views expressed are those of the authors and do not necessarily reflect any position or policy of the U.S. Nuclear Regulatory Commission.

REFERENCES

1. Driscoll, K., "A View from the Trenches," Dependable Software Technology Exchange, Software Engineering Institute, Pittsburgh, PA, March 1994.

2. Wika, K. G., "A User Interface and Control Algorithm for the Video Tumor Fighter," Masters Thesis, University of Virginia, May 1991.

3. Wing, J., "A Specifiers Introduction to Formal Methods", *IEEE Computer*, Vol. 23, No. 9, September 1990.

4. Henninger, K, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application," *IEEE Transactions on Software Engineering*, Vol. SE6, No. 1, January 1980, pp. 2-13.

5. Defense Standard 0056, *Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defense Equipment*, Ministry of Defense, United Kingdom, 1992.