N89-16301 167046 7P.

# A Computer-Based Specification Methodology

Robert G. Munck

The MITRE Corporation
Bedford, MA 01730
*Munck@MITRE-Bedford.ARPA*

## ABSTRACT

It is becoming clear that our standard way of writing specifications -- requirements, design, test, and other types -- is inadequate for large, complex, and long-lived systems. The process by which they are created is unstructured and often cursory, and the resulting paper documents are bulky, vague, inconsistent, and difficult to publish, distribute, and update. We are especially bad at writing understandable, consistent, and sufficiently-detailed requirements specifications.

Part of the problem comes from the shortcomings of written English and the essentially *linear* sentence/paragraph/chapter structure of specifications; it has been aggravated by widespread use of word processors that support nothing but text. Text will always be a part of specifications, but there are other forms of expression that can be more suitable for other parts: data-flow, SADT[ROSS75], and Buhr[BUHR84] diagrams, non-linear text or "hypertext,"[VAN DAM70] spreadsheet-supported tables, charts, and graphs, animation of algorithms and procedures, geometric modeling, and voice and video processing. All of these become viable possibilities in the high-capacity, display-oriented workstations of the proposed Space Station Data Management System.

The use of exotic, "high-tech" presentation media in specifications will not automatically make them easy to produce and understand; it is more important that there be a **methodology** for creating them that emphasizes correctness and clarity of presentation, and which supports cooperative work over a network. The most complete and mature such methodology is SofTech's SADT™. SADT is unique in the amount of attention it pays to the way people work together and as individuals and in its facilities for specifying requirements independent of any particular implementation.

SADT's most widely-used component is the hierarchical box-and-arrow diagram notation. In the full methodology, that notation is supported by an "infrastructure" of procedures, formats, protocols, and "ways of thinking" that make it possible for many people to *work together* on a large project. For example, the **Reader/Author Cycle** is a peer review procedure that emphasizes constructive criticism and a disciplined exchange of ideas. **Reader Kits** and their associated **Kit Files** provide a mechanism for working on part of a specification without losing sight of its relationship to the whole AND for tracking the evolution of the specification over time.

The paper proposes a network-based system for writing, reviewing, and publishing multi-media specifications with tools and procedures based on SADT methodology. It envisions people at universities, companies, and NASA sites all over the world working together to prepare a requirements or design specification and discusses the possibility of semi-automatic conversion of such a specification to Ada[1] code, currently under investigation at MITRE. The computer-based SADT tools developed in that project will be described.

## 1 Everybody's Talking

Natural language has evolved over the millennia as our most powerful tool, that which truly separates us from animals. However, it is becoming apparent that "written English" using traditional forms and media (chapters and paragraphs, paper and ink) is insufficient to communicate the very large, complexly-interrelated concepts of a modern computer-based electronic system. To put it another way, our

---

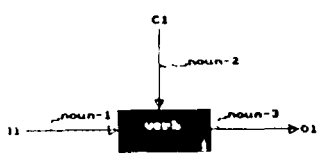
1. Ada is a registered trademark of the United States Government (Ada Joint Program Office)

specification documents vary from "unsatisfactory" to "horrible" in doing what they are supposed to do.

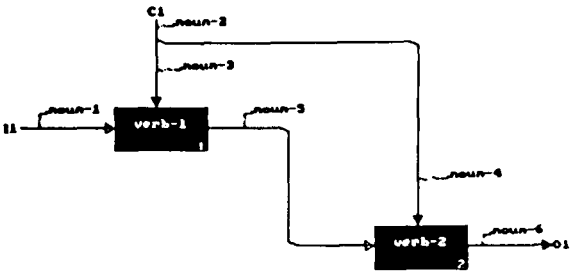
### 1.1 Words in a Row

Written English (or the language of your choice) is essentially the spoken form translated from sound to ink patterns on a word-by-word basis and laid out in linear form. As such, it makes many of the same assumptions as speech; for example, pronouns are based on the assumption that the listener has a high-speed short-term memory that can match a pronoun to the last object named, thus reducing repetition of names. Unfortunately, in technical writing the use of pronouns is overwhelmed by the great number of objects needing to be named. Names of things are therefore repeated over and over, and are often long proper noun phrases containing several capitalized adjectives and adverbs. The resulting text is much like a road containing frequent potholes and boulders.

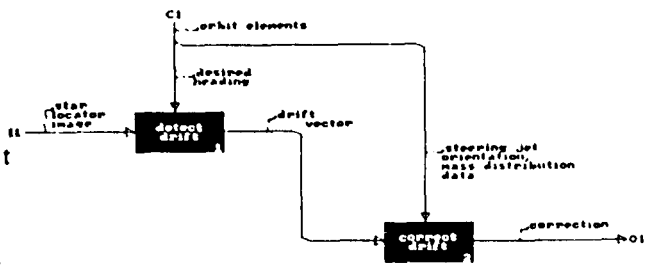
Graphical languages such as SADT boxes-and-arrows and Buhr diagrams reduce the need to repeat names by using two dimensions instead of the linear one-dimensional representation of written text. That is, an object can be associated with things above, below, right, and left of it, not just to the left. SADT also generalizes the *noun - verb - object* sentence structure of English into two dimensions, thus better approximating the way people think. In a simple example, the SADT fragment




is the equivalent of the English "*noun-1* is *verb*ed to make *noun-3* as controlled by *noun-2*". A more complex example, which demonstrates the generalization, is



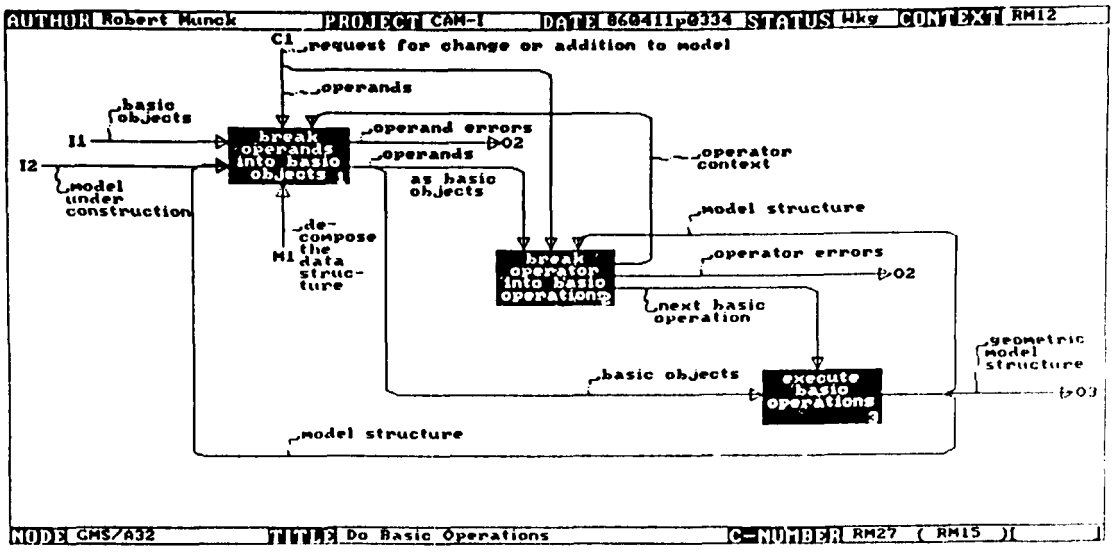

which says "*noun-1* is *verb-1*'ed to make *noun-5* as controlled by the *noun-3* aspect of *noun-2*; *noun-5* is *verb-2*'ed to make *noun-6* as controlled by the *noun-4* aspect of *noun-2*". A somewhat more real example:




Put into English, it says (approximately):

> The star locator image is used to detect drift by using the desired heading from the orbit elements to calculate a drift vector. The drift vector is then used to compute a correction from it and the steering jet and mass distribution data from the orbit elements data.

A complete SADT diagram, such as this:




Might translate into a page or more of normal text. That text would be as readable or unreadable as text specifications normally are.

The diagrams shown above all have activities, actions, or verbs in their boxes and data, things, or objects as their arrows. There are in fact four kinds of diagrams, of which these are only one, called the **Activity Diagram**. There are also **Data Diagrams, State Diagrams,** and **Transition Diagrams**. Data diagrams, in which the arrows are activities, are similar to those drawn in Data Base design and Object-Oriented Design. State and Transition diagrams are useful in real-time systems.

## 1.2 What It's All About

Human communication is based on the fact that we have a vocabulary in common with each other. Unfortunately, that commonality is only approximate; we can never be entirely certain what someone else means by "red" or "big" or "Multi-modal Phased Radar Array Serial Interface." Technical specifications can be characterized as a semi-ordered set of terms and the definitions of those terms -- in the ultimate the entire specification is a definition of its title. The problems that arise include multiple and conflicting definitions, the definition of a term being "far away" from its usage and difficult to find, and multiple terms having the same definition.

Definitions of terms often themselves contain terms that need definition. SADT uses the hierarchy resulting from this as its organizational backbone. Each box on a diagram contains a word or term that has some meaning to the author of the diagram; it may have a different meaning to a reader of it. If the author feels that readers might have a different meaning for a box than his intent or not know what it means, he creates a new (child) diagram that "explains" or "defines" the box in greater detail. Boxes on the child diagram that need further explanation are themselves expanded into diagrams, until all terms in all unexpanded boxes are in common parlance and unambiguous.

One of the strengths of natural language is that words can have different meanings in different contexts. This reduces by several orders of magnitude the number of different words we need. However, specification writers often attempt to give certain important words rigid definitions for all contexts, placing those definitions in a glossary. Those reading the specification must, in effect, memorize the

entire glossary for the duration of their reading; otherwise they will have to flip back and forth to it continuously, with no way to know if they need to look up a particular word. In SADT, there is an indication on each box if it is expanded. Boxes on different diagrams containing the same word or phrase may have the same or different expansions. This is the SADT equivalent of the common notion that a word or phrase may have different meanings when used in different contexts.

### 1.3 SADT Media and the Message

SADT was originally designed for use with no computer support; a team having standard office supplies and a copier could create very large, very high-quality specifications. In fact, users tended to resist having their diagrams even typed or typeset; a diagram produced with a good pen, a straight-edge or flowchart template (for the curved corners), and legible handwriting seemed more "comfortable." An early attempt to computerize the production of diagrams using a time-shared mainframe[SMITH81] was unsatisfactory due to slow response time.

Despite the power of the SADT filing and archive system (discussed later), large and long-term projects found the maintenance of a large set of diagrams (thousands) to be burdensome. Fortunately, the personal computer has now become powerful enough to support SADT, and in fact is proving to be an extremely valuable addition. There are at least four announced or planned SADT systems on the market.[SAIB85]

The SAda[MUNCK85] system, implemented by the author as an IR&D project at MITRE, runs on an IBM PC or equivalent. A satisfactory system with the necessary graphics and telecommunications can be bought for $2500 hardware costs; a "super" system with a big color display and laser printer might cost $10,000.

## 2 The Way We Work

The above discussion has shown a few of the many ways that SADT makes it possible to have a readable, understandable technical specification. In general, it does so by relaxing or generalizing English grammar, sentence and paragraph structure, and the division into sections, appendices, glossaries, annexes, and volumes of normal specifications. With SADT, the most complex systems that we are capable of building can be specified understandably. Among the most complex system specified in SADT to date is the financial system of the Department of Energy. The complete specification took more than 25 analyst-years to write and, printed double-sided, was over two feet thick. Because it was done on paper before computer support was available, the document is quite intimidating by its sheer mass, but still vastly preferable to a text equivalent.

Of course, there is no free lunch. Specifying a complex system well with SADT takes a great deal of hard work by trained, experienced, smart people. That work is made as productive as possible by other features of SADT that deal with the way people work together and individually. These features might be called the "management" or "sociological" aspects of SADT.

### 2.1 All Together Now

The creation of specifications is usually a quite chaotic process in most organizations. A common feature is the "brainstorm session" at which a number of people present ideas, argue, and fill blackboards with scribbling. At the end, several participants are charged with "writing up the results." However, they will capture only the last set of ideas proposed and not rejected; other good ideas disappear forever the next time the blackboard is erased or never appear because their conceiver is absent or doesn't communicate well in noisy meetings. The basic idea of brainstorming is good: communicating "half-baked" ideas quickly to others who can grab the good ones and add their own improvements. We need a better process and medium than the noisy meeting and blackboard.

SADT includes the Reader/Author Cycle to replace this aspect of writing specifications. It works as follows:

1. One analyst, called the *Author*, creates a small number of diagrams. His SADT training tells him to limit the diagrams to one major thought or amount of information, approximately a half-day's work on his part to create and an hour's work to read. This amount is typically one parent diagram and three to five child diagrams. The SAda drawing tool helps him create the diagrams using a mouse and keyboard such that his thinking is about the subject matter, not the mechanics of drawing; there is no need to sketch diagrams on paper and enter them into the computer as a separate step.

2. The Author assembles these diagrams into a *Reader Kit* and sends the kit to a small number (1-4) of his colleagues, called *Readers*. These diagrams are transmitted by electronic mail and appear in a "to-be-read" directory in the Readers' machines.

3. Each Reader reads the kit within one working day. He writes comments on the diagrams with arrows and circles indicating where they apply, using the mouse and keyboard. SADT Readers are trained at great length to make their comments constructive and non-threatening; in effect, there is a "code of courtesy" for writing comments. Note that the Author does not have a large "psychic investment" in the diagrams; he has spent a relatively short amount of time creating them. This contrasts to the difficulty of criticizing something that someone has spent weeks or months producing. Readers who are also trained to be Authors comment on the format and understandability of the diagrams as well as their technical content.

4. The Reader transmits his comments back to the Author.

5. The Author reads the comments from each Reader within one working day and writes a reply to each one. Here again, the Author is trained to write replies that are constructive and helpful, not argumentative. While doing this, he also makes notes on the diagrams indicating changes to be made that the comments have inspired. Comments, replies, and notes are overlays or windows that can be added and removed from the diagram on the display; on a color display, they appear in color.

6. The Author transmits each Reader's comments back to him.

7. The Reader reads the replies and adds additional notes of his own. The diagrams, comments, replies, and notes are added to his files.

8. If necessary, the Author revises his diagrams and sends them out again, starting another cycle. This time, however, the Readers have the previous revision with the comments and replies. They can therefore check that problems they noticed have been fixed.

The Cycle is "kept going" in the manual system by the Librarian, a clerk trained in SADT procedures. He does the mechanical tasks such as copying and filing, and makes sure that the participants do their jobs in the time allowed. In the computer-based system, no Librarian is needed, and the participants may be far apart physically on a loosely-coupled network.

The Reader/Author Cycle has been shown to be an extremely powerful organizing influence on technical work of all kinds. Many organizations that were exposed to it through SADT training now use it for most or all of their work, even when other aspects of SADT are not involved. It appears to be a good match to the needs and organization of NASA.

When used with the "code of courtesy" and other aspects of SADT, the Cycle brings out the best, most creative thoughts of the participants, reduces conflict, and captures the processes by which decisions are made, not just their results. People who have worked on successful SADT projects tend

to urge others to use it with the fervor of religious converts.

### 2.2 In Organization There is Strength

We have mentioned the two-dimensional aspect of diagrams and later a third dimension, that of expansion of boxes into diagrams. There is also a dimension of time, in which each diagram has a pointer to the diagram that it replaced and to the one that replaced it, and notes by the author explaining why it was replaced. The result is a fairly complex data structure, but one that proves easy to navigate with the right computer support.

A single set of diagrams related hierarchically, starting from a single "top level" diagram, is called a model. A model is a top-down exposition of a single aspect or part of the system as seen from a single, stated viewpoint. For example, we might have models of a single instrument from viewpoints such as a user, a maintenance technician, a programmer, a telemetry system, and a power system. Each of these models will emphasize the parts that are important from its given viewpoint and will have pointers to other models for other parts and to models of other aspects of the system to which it is related.

## 3 Make it Run

As done on paper, an SADT specification can be an extremely readable document, leading to much better implementation. In the computer-based system, there are even more possibilities:

- A model of the activities of a project, with estimated time and manpower attached to each box, can be analyzed by the machine to determine a schedule and indicate which activities are on the critical path. This project model can be maintained by the program office as the master project schedule, with pointers from each box to the current status report for that activity. One would be able to review progress informally and conveniently by browsing through the model.

- A model of a piece of software can have execution time and resource use estimates attached to each box. It can then be "executed" as a simulation to predict performance.[BUCHERT81] The simulator could "animate" the model on a graphic display as it executes. Small meters or bar charts could be attached to boxes and arrows on the display to show current values such as processing rate, queue length, frequency, and values of variables.

- A detailed model of a piece of software can be converted into skeleton Ada[Ada83] code defining the task structure. Each lowest-level box can then be coded by an Ada programmer (or the appropriate function found in a library) and combined with the skeleton to make a running system. The SAda project at MITRE is beginning to explore this possibility.

- A model could be connected to its implementation, hardware or software, by diagnostic or metering probes. It could then "run" in the same way that the simulator animation discussed above did. A person monitoring the system could move up and down between levels of detail.

- A detailed model might be able to be converted mechanically into a custom integrated circuit or piece of wafer-scale integration. This model might also have run as a simulation or been converted into running Ada code.

Most of the above suggestions have been tried in one way or another, and all showed promise. The time is ripe to begin work on an *infrastructure* or *support environment* on which the tools for writing, reading, and "exercising" computer-based specifications can be integrated. It is clear that such spec-writing support environments would have a great deal in

common with programming support environments, to the point of both being part of a single larger system.

## 4 Conclusion

Standard practices for creating and using system specifications are inadequate for large, advanced-technology systems. We need to break away from paper documents in favor of documents that are stored in computers and which are read and otherwise used with the help of computers. An SADT-based system, running on the proposed Space Station data management network, could be a powerful tool for doing much of the required technical work of the Station, including creating and operating the network itself.

## References


[Ada83] U.S. Department of Defense **Reference Manual for the Ada Programming Language**, ANSI/MIL-STD-1815A-1983.

[BUCHERT81] Buchert, R.F., K. H. Evers, and P. R. Santucci, "SADT/SAINT Simulation Technique," *National Aerospace and Electronics Conf. Proc.*, 1981.

[BUHR84] Buhr, R.J.A, **System Design With Ada**, Prentice Hall, Englewood Cliffs, NJ, 1984

[COMBELIC78] Combelic,D., "User Experience with New Software Methods (SADT)," *Proc. NCC*, Vol. 47, 1978, pp. 631-633.

[MUNCK85] Munck, R, "Toward Large Software Systems that Work," *AIAA/ACM/NASA/IEEE Computers in Aerospace V Proc.*, Oct. 21-24, 1985.

[ROSS75] Ross, D., et al, **SADT Structured Analysis and Design Technique Author Guide**, SofTech, Inc. 6490-1, October, 1975, Waltham, MA.

[SAIB85] Saib, S., "A Life-Cycle Environment," *AIAA/ACM/NASA/IEEE Computers in Aerospace V Proc.*, Oct. 21-24, 1985.

[SMITH81] Smith, D.G., "Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II -- Automated IDEF-0 Development," NTIS B062454-B052459, August, 1981.

[VAN DAM70] van Dam, A, and D.E. Rice, "Computers and Publishing: Writing, Editing, and Printing," **Advances in Computers**, Academic Press, New York, 1970.


**Biography**

**Robert Munck** received an AB in computer science from Brown University. In twenty years in the field, he has taught at Brown and worked at SofTech, Prime Computer, the Naval Research Lab, and MITRE, where he is presently writing a CAIS operating system in Ada for the Intel 80386.