Structured Analysis (SA): A Language for Communicating Ideas

DOUGLAS T. ROSS

Abstract—Structured analysis (SA) combines blueprint-like graphic language with the nouns and verbs of any other language to provide a hierarchic, top-down, gradual exposition of detail in the form of an SA model. The things and happenings of a subject are expressed in a data decomposition and an activity decomposition, both of which employ the same graphic building block, the SA box, to represent a part of a whole. SA arrows, representing input, output, control, and mechanism, express the relation of each part to the whole. The paper describes the rationalization behind some 40 features of the SA language, and shows how they enable rigorous communication which results from disciplined, recursive application of the SA maxim: "Everything worth saying about anything worth saying something about must be expressed in six or fewer pieces."

Index Terms—Graphic language, hierarchic, requirements analysis, requirements definition, structured analysis (SA), structured programming, system analysis, system design, top-down.

I. Blueprint Language

NEITHER Watt's steam engine nor Whitney's standardized parts really started the Industrial Revolution, although each has been awarded that claim, in the past. The real start was the awakening of scientific and technological thoughts during the Renaissance, with the idea that the lawful behavior of nature can be understood, analyzed, and manipulated to accomplish useful ends. That idea itself, alone, was not enough, however, for not until the creation and evolution of blueprints was it possible to express exactly how power and parts were to be combined for each specific task at hand.

Mechanical drawings and blueprints are not mere pictures, but a complete and rich language. In blueprint language, scientific, mathematical, and geometric formulations, notations, mensurations, and naming do not merely describe an object or process, they actually model it. Because of broad differences in subject, purpose, roles, and the needs of the people who use them, many forms of blueprint have evolved, but all rigorously present well structured information in understandable form.

Failure to develop such a communication capability for data processing is due not merely to the diversity and complexity of the problems we tackle, but to the newness of our field. It has naturally taken time for us to escape from naive "programming by priesthood" to the more mature approaches, such as structured programming, language and database design, and software production methods. Still missing from this expanding repertoire of evidence of maturity, however, is the common thread that will allow all of the pieces to be tied together into a predictable and dependable approach.

Manuscript received June 21, 1976; revised September 16, 1976.
The author is with SofTech, Inc., Waltham, MA 02154.

II. Structured Analysis (SA) Language

It is the thesis of this paper that the language of structured analysis (SA), a new disciplined way of putting together old ideas, provides the evolutionary natural language appropriate to the needs of the computer field. SA is deceptively simple in its mechanics, which are few in number and have high mnemonic value, making the language easy and natural to use. Anybody can learn to read SA language with very little practice and will be able to understand the actual information content being conveyed by the graphical notation and the words of the language with ease and precision. But being a language with rigorously defined semantics, SA is a tough taskmaster. Not only do well conceived and well phrased thoughts come across concisely and with precision, but poorly conceived and poorly expressed thoughts also are recognized as such. This simply *has* to be a fact for any language whose primary accomplishment is valid communication of understanding. If both the bad and the good were *not* equally recognizable, the understanding itself would be incomplete.

SA does the same for any problem chosen for analysis, for every natural language and every formal language are, by definition, included in SA. *The only function of SA is to bind up, structure, and communicate units of thought expressed in any other chosen language.* Synthesis is composition, analysis is decomposition. *SA is structured decomposition, to enable structured synthesis to achieve a given end.* The actual building-block elements of analysis or synthesis may be of any sort whatsoever. Pictures, words, expressions of any sort may be incorporated into and made a part of the structure.

The facts about Structured Analysis are as follows.

1) It incorporates any other language; its scope is universal and unrestricted.

2) It is concerned only with the orderly and well-structured decomposition of the subject matter.

3) The decomposed units are sized to suit the modes of thinking and understanding of the intended audience.

4) Those units of understanding are expressed in a way that rigorously and precisely represents their interrelation.

5) This structured decomposition may be carried out to any required degree of depth, breadth, and scope while still maintaining all of the above properties.

6) Therefore, SA greatly increases both the quantity and quality of understanding that can be effectively and precisely communicated well beyond the limitations inherently imposed by the imbedded natural or formal language used to address the chosen subject matter.

The universality and precision of SA makes it particularly effective for requirements definition for arbitrary systems problems, a subject treated in some detail in a companion pa-

per (see [5]). Requirements definition encompasses all aspects of system development prior to actual system design, and hence is concerned with the discovery of real user needs and communicating those requirements to those who must produce an effective system solution. Structured Analysis and Design Technique (SADT™) is the name of SofTech's proprietary methodology based on SA. The method has been applied to a wide range of planning, analysis, and design problems involving men, machines, software, hardware, database, communications procedures, and finances over the last two years, and several are cited in that paper. It is recommended that that paper (see [5]) be read prior to this paper to provide motivation and insight into the features of SA language described here.

SA is not limited to requirements definition nor even problems that are easily recognized as system problems. The end product of an SA analysis is a working model of a well-structured understanding, and that can be beneficial even on a uniquely personal level—just to "think things through." Social, artistic, scientific, legal, societal, political, and even philosophic subjects, all are subject to analysis, and the resulting models can effectively communicate the ideas to others. The same methods, approach, and discipline can be used to model the problem environment, requirements, and proposed solution, as well as the project organization, operation, budget, schedule, and action plan. Man thinks with language. Man communicates with language. SA structures language for communicating ideas more effectively. *The human mind can accommodate any amount of complexity as long as it is presented in easy-to-grasp chunks that are structured together to make the whole.*

III. Outline of the Demonstration

Five years ago I said in an editorial regarding software [1]: "Tell me *why* it works, not *that* it works." That is the approach taken in this paper. This paper does not present a formal grammar for the SA language—that will come later, elsewhere. This paper also is not a user manual for either authors or readers of the language—a simple "how to" exposition. Instead, we concentrate here on the motivation *behind* the features of SA in an attempt to convey directly an appreciation for its features and power even beyond that acquired through use by most SA practitioners. SA has been heavily developed, applied, taught, and used for almost three years already, but the design rationale behind it is first set down here.

SA (both the language and the discipline of thought) derives from the way our minds work, and from the way we understand real-world situations and problems. Therefore, we start out with a summary of principles of exposition—good storytelling. This turns out to yield the familiar top-down decomposition, a key component of SA. But more than that results, for consideration of how we view our space-time world shows that we always understand anything and everything in terms of *both* things *and* happenings. This is why all of our languages have *both* nouns *and* verbs—and this, in turn, yields the means by which SA language is universal, and can absorb any other language as a component part.

SA supplies rigorous structural connections to any language whose nouns and verbs it absorbs in order to talk about things and happenings, and we will spend some time covering the basics carefully, so that the fundamentals are solid. We do this by presenting, in tabular form, some 40 basic features, and then analyzing them bit by bit, using SA diagrams as figures to guide and illustrate the discussion.

Once the basics have thus been introduced, certain important topics that would have been obscure earlier are covered in some depth because their combinations are at the heart of SA's effectiveness. These topics concern constraints, boundaries, necessity, and dominance between modular portions of subject matter being analyzed. It turns out that constraints based on purpose and viewpoint actually *make* the structure. The depth of treatment gives insight into how we understand things.

The actual output of SA is a hierarchically organized structure of separate diagrams, each of which exposes only a limited part of the subject to view, so that even very complex subjects can be understood. The structured collection of diagrams is called an *SA model*. The demonstration here concludes with several special notations to clarify presentation and facilitate the orderly organization of the material. Since actual SA diagrams (some good, some illustrating poor style) are used, as figure illustrations, the reader is exposed here to the style of SA even though the SA model represented by the collection of figures is not complete enough to be understandable by itself. Later papers will treat more advanced topics and present complete examples of SA use and practice in a wide variety of applications.

IV. Principles of Good Storytelling

There are certain basic, known principles about how people's minds go about the business of understanding, and communicating understanding by means of language, which have been known and used for many centuries. No matter how these principles are addressed, they always end up with hierarchic decomposition as being the heart of good storytelling. Perhaps the most relevant formulation is the familiar: "Tell 'em whatcha gonna tell 'em. Tell 'em. Tell 'em whatcha told 'em." This is a pattern of communication almost as universal and well-entrenched as Newton's laws of motion. It is the pattern found in all effective forms of communication and in all analyses of why such communication is effective. Artistic and scientific fields, in addition to journalism, all follow the same sequence, for that is the way our minds work.

Only something so obvious as not to be worth saying can be conveyed in a single stage of the communication process, however. In any worthwhile subject matter, Stage Two ("Tell 'em") requires the parallel introduction of several more instances of the same pattern starting again with Stage One. Usually a story establishes several such levels of telling, and weaves back and forth between them as the need arises, to convey understanding, staying clear of excesses in either detail (boredom) or abstraction (confusion).

™Trademark of SofTech, Inc.

V. The SA Maxim

This weaving together of parts with whole is the heart of SA. The natural law of good communications takes the following, quite different, form in SA:

"Everything worth saying
about anything worth saying something about
must be expressed in six or fewer pieces."

Let us analyze this maxim and see how and why it, too, yields hierarchically structured storytelling.

First of all, there must be something (anything) that is "worth saying something about." We must have some subject matter that has some value to us. We must have an interest in some aspect of it. This is called establishing the *viewpoint* for the model, in SA terminology. Then we must have in mind some audience we want to communicate with. That audience will determine what is (and is not) "worth saying" about the subject from that viewpoint. This is called establishing the *purpose* for the model, in SA terminology. As we will see, every subject has many aspects of interest to many audiences, so that there can be many viewpoints and purposes. But each SA model must have only one of each, to bound and structure its subject matter. We also will see that each model also has an established *vantage point* within the purpose-structured context of some other model's viewpoint, and this is how multiple models are interrelated so that they collectively cover the whole subject matter. But a single SA model considers only worthy thoughts about a single worthy subject.

The clincher, however, is that *every* worthy thought about that worthy subject must be included. The first word of the maxim is *everything*, and that means exactly that—absolutely nothing that fits the purpose and viewpoint can be left out. The reason is simple. By definition everything is the subject itself, for otherwise it would not *be* that subject—it would be a *lesser* subject. Then, if the subject is to be broken into six or fewer pieces, every single thing must go into exactly one of those (nonoverlapping) pieces. Only in this way can we ensure that the subject stays the same, and does not degenerate into some lesser subject as we decompose it. Also, if overlapping pieces were allowed, conflicts and confusions might arise.

A "piece" can be anything we choose it to be—the maxim merely requires that the single piece of thought about the subject be broken into several (not too few, and not too many[1]) pieces. Now, certainly if the original single piece of thought about the subject is worthy, it is very unlikely that the mere breaking of it into six-or-fewer pieces exhausts that worth. The maxim still applies so that every one of them must similarly be expressed in six-or-fewer *more* pieces—again and again—until the number of pieces has grown to suit the total worthiness. At a fine enough level of decomposition, it is not worth continuing. No further decomposition is required for completely clear understanding. Thus we see that the SA maxim must be interpreted *recursively*, and yields top-down hierarchic decomposition. The SA language allows this hierarchic structure to be expressed (see Fig. 1).

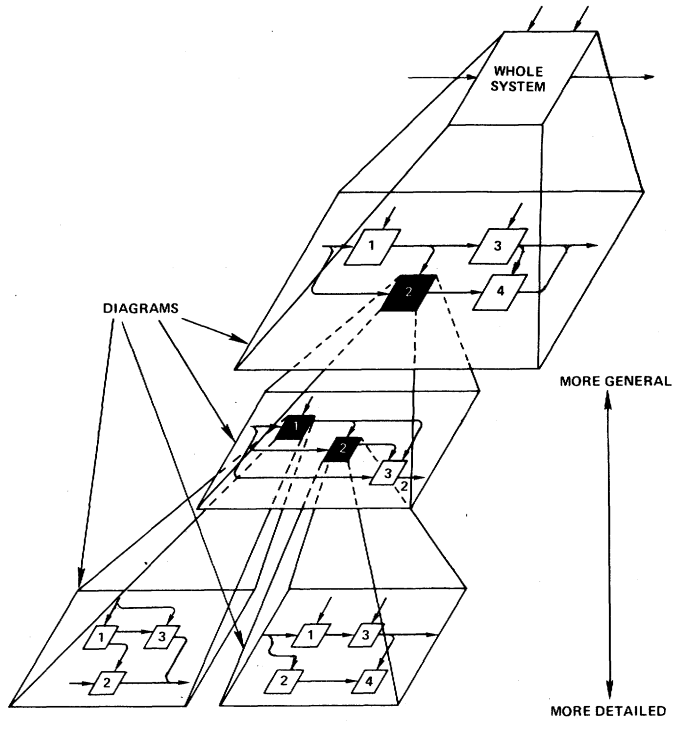

Fig. 1. Structured decomposition.

VI. Expression

In the maxim, the word "express" covers both the rigorous grammar of SA language itself, as well as the grammar (however well or ill formed) of the natural language chosen to address the subject matter. By definition, SA language includes all other languages, and regardless of what language is embedded, the decomposition discipline (expressed by the SA language component of the combined language) ensures that at each stage, the natural language (whatever it may be) is used to address and express only every worthy thought about a more and more restricted piece of the worthy subject matter. Because of this orderly zeroing-in, SA certainly cannot decrease the effectiveness of that chosen language. In effect, the SA maxim is valid by definition, for whenever the subject matter has already been broken down to such a fine level that the SA decomposition would add nothing to what already would be done (as, for example, in jokes or some poetry) the chosen language stands by itself, not decreased in effectiveness.

Most of the time the conscious practice of Structured Analysis and its thought discipline improves people's ability to think clearly and find solutions. In the cases where this does not happen, however, Structured Analysis still "works," in the sense that the bad portions stand out more clearly and are understood to be bad and needing further attention. For the next step in our demonstration we consider thoughts, and the expression of thoughts in language.

[1] Many people have urged me to relate the magic number "six" to various psychological studies about the characteristics of the human mind. I won't. It's neither scientific nor "magic." It is simply the *right* number. (Readers who doubt my judgement are invited to read for themselves the primary source [6].) The only proper reference would be to the little bear in the Goldilocks story. His portions always were "just right," too!

VII. Things and Happenings

We live in a space-time world. Numerous philosophical and scientific studies, as well as the innate experience of every person, shows that we never have any understanding of any subject matter except in terms of our own mental constructs of "things" and "happenings" of that subject matter. It seems to be impossible for us to think about anything without having that subject automatically be bounded in our minds by a population of parts or pieces (concrete or abstract–but in any case "nominal" things, i.e., literally things to which we give names or nouns) which provide the basis for our even conceiving of the subject matter as a separate subject. Immediately, however, once we are aware of the things, we are aware of the happenings–the relationships, changes, and transformations which take place between and among those things and make the subject matter meaningful or interesting (the "verbial" things, to which we give action words or verbs).

The universality of things and happenings provides the next basic step of decomposition (after the still more fundamental decomposition of recognizing and isolating the purpose and viewpoint which established the "worth" of possible things to say about the "worth" of the subject matter). Every one of our languages, whether natural or artificial, formal or informal, has those two complementary aspects–nouns and verbs, operators and operands, etc.–to permit the expression of thoughts about the subject matter. Thus the means is provided to incorporate any other language into SA. The incorporation of other languages into SA is not forced, nor awkward.

SA language provides the same graphic notation for both the things and the happenings aspects of any subject. Every SA model has two dual aspects–a *thing* aspect, called the *data decomposition*, and a *happening* aspect, called the *activity decomposition*. The model is incomplete without both decompositions.

VIII. Bounded Subject Matter

So we have now established the starting premises. The SA maxim forces gradual, top-down decomposition, leaving nothing out at any stage, and matching good storytelling exposition. The things and happenings (*data* and *activities*, in SA technical terms) match the nominal and verbial construction of any chosen language for directly addressing the subject, so we will never be "at a loss for words." Now we are ready to address the specifics–how SA language (mostly graphical, using boxes and arrows) actually allows well structured expression of well structured thought. We do this in stages: 1) we dump the entire body of the subject matter all at once into a table of some 40 separate items of notation and conventions–just to bound the subject itself; 2) we then start to pick our way through these topics, starting with those that define the basics of boxes and arrows; and 3) then we will use those basic expository capabilities to complete the consideration of the list.

In a prior, companion paper [2], which had its roots in the same background that led to the development of SA, we described and illustrated a univeral, standard pattern or process which appears to permeate all of software engineering and problem-solving in general. Since that pattern is so close to the natural phenomena of understanding which we are discussing here with respect to SA itself, we will use it to motivate, clarify, and structure the presentation. The idea of the pattern is captured in five words: 1) purpose; 2) concept; 3) mechanism; 4) notation; and 5) usage. Any systematic approach to a problem requires a concise purpose or objective that specifies which aspect of the problem is of concern. Within that purpose we formulate a valid conceptual structure (both things and happenings) in terms of which the problem can be analyzed and approached. We then seek out (or work out) the designs (mechanisms–concrete or abstract, but always including both data and activity aspects) which are capable of implementing the relevant concepts and of working together to achieve the stated purpose. (This combines three of the five words together.) Now, purpose, concepts, and mechanism, being a systematic approach to a *class* of problems, require a notation for expressing the capabilities of the mechanism and invoking its use for a particular problem in the class. Finally, usage rules are spelled out, explicitly or by example, to guide the use of the notation to invoke the implementation to realize the concept to achieve the specified purpose for the problem. The cited paper [2] gives numerous carefully drawn examples showing how the pattern arises over and over again throughout systematic problem solving, at both abstract and concrete levels, and with numerous hierarchic and cross-linked interconnections.

IX. The Features of SA Language

Fig. 2 is a tabulation of some 40 features or aspects of SA which constitute the basic core of the language for communication. For each feature, the purpose, concept, mechanism, and notation are shown. Usages (for the purposes of this paper) are covered only informally by the examples which follow. The reader should scan down the "purpose" column of Fig. 2 at this time, because the collection of entries there set the objectives for the bounded subject matter which we are about to consider. Note also the heavy use of pictures in the "notation" column. These are components of graphic language. But notice that most entries mix *both* English *and* graphic language into a "phrase" of SA notation. Clearly, any other spoken language such as French, German, or Sanskrit could be translated and substituted for the English terms, for they merely aid the understanding the syntax and semantics of SA language itself.

In Fig. 2, the *name* and *label* portions of the "notation" column for rows 1 and 2, and the corresponding *noun* and *verb* indications in rows 6 and 7 are precisely the places where SA language absorbs other natural or formal languages in the sense of the preceding discussion. As the preceding sections have tried to make clear, *any* language, whether informal and natural or formal and artificial, has things and happenings aspects in the nominal and verbial components of its vocabulary. These are to be related to the *names of boxes* and *labels on arrows* in order to absorb those "foreign" languages into SA language.

Notice that it is not merely the nouns and verbs which are absorbed. Whatever richness the "foreign" language may possess, the full richness of the nominal and verbial expressions, including modifiers, is available in the naming and labeling por-

	PURPOSE	CONCEPT	MECHANISM	NOTATION	NODE
1	BOUND CONTEXT	INSIDE/OUTSIDE	SA BOX	NAME	A11
2	RELATE/CONNECT	FROM/TO	SA ARROW	LABEL	A12
3	SHOW TRANSFORMATION	INPUT-OUTPUT	SA INTERFACE	INPUT / OUTPUT	A13
4	SHOW CIRCUMSTANCE	CONTROL	SA INTERFACE	CONTROL	A14
5	SHOW MEANS	SUPPORT	SA MECHANISM	MECHANISM	A15
6	NAME APTLY	ACTIVITY: HAPPENINGS / DATA: THINGS	SA NAMES	ACTIVITY: VERB / DATA: NOUN	A211
7	LABEL APTLY	THINGS / HAPPENINGS	SA LABELS	NOUN / VERB	A212
8	SHOW NECESSITY	I-O / C-O	PATH		A213
9	SHOW DOMINANCE	C / I	CONSTRAINT		A214
10	SHOW RELEVANCE	ICO / ICO	ALL INTERFACES		A215
11	OMIT OBVIOUS	C-O / I-O	OMITTED ARROW		A216
12	BE EXPLICIT WITHOUT CLUTTER	PIPELINES, CONDUITS, WIRES	BRANCH	A / A, A	A221
13			JOIN	A, A / A	A221
14	BE CONCISE AND CLEAR	CABLES, MULTI-WIRES	BUNDLE	A, B / C(=AUB)	A222
15			SPREAD	C=(AUB) / A, B	A222
16	SHOW EXCLUSIVES	EXPLICIT ALTERNATIVES	OR BRANCH	A OR B / A, B	A223
17			OR JOIN	A, B / A OR B	A223
18	SHOW INTERFACES TO PARENT DIAGRAM	PARENT / CHILD / ARROWS PENETRATE	SA BOUNDARY ARROWS (ON CHILD)	NO BOX SHOWN	A231
19	SHOW EXPLICIT PARENT CONNECTION	NUMBER CONVENTION FOR PARENT, WRITE ICOM CODE ON CHILD BOUNDARY ARROWS		c1, c2 / o2 (ON CHILD)	A232
20	SHOW UNIQUE DECOMPOSITION	DETAIL REFERENCE EXPRESSION (DRE)	C-NUMBER OR PAGE NUMBER OF DETAIL DIAGRAM	BOX / X → DRE	A233
21	SHOW SHARED OR VARIABLE DECOMPOSITION	DRE WITH (MODEL NAME)	SA CALL ON SUPPORT	BOX / STUB → X → DRE	A234

	PURPOSE	CONCEPT	MECHANISM	NOTATION	NODE
22	SHOW COOPERATION	INTERCHANGE OF SHARED RESPONSIBILITY	SA 2-WAY ARROWS		A311
23	SUPPRESS INTERCHANGE DETAILS	ALLOW 2-WAY WITHIN 1-WAY PIPELINES	2-WAY TO 1-WAY BUTTING ARROWS		A312
24	SUPPRESS "PASS-THROUGH" CLUTTER	ALLOW ARROWS TO GO OUTSIDE DIAGRAMS	SA "TUNNELING" (WITH REFERENCES)	PARENT OFFSPRING	A313
25	SUPPRESS NEEDED-ARROW CLUTTER	ALLOW TAGGED JUMPS WITHIN DIAGRAM	TO ALL or FROM ALL	TO ALL	A314
26	SHOW NEEDED ANNOTATION	ALLOW WORDS IN DIAGRAM	SA NOTE	NOTE:	A32
27	OVERCOME CRAMPED SPACE	ALLOW REMOTE LOCATION OF WORDS IN DIAGRAM	SA FOOTNOTE	(n=integer) n words n	A32
28	SHOW COMMENTS ABOUT DIAGRAM	ALLOW WORDS ON (NOT IN) DIAGRAM	SA META-NOTE	n (n=integer)	A32
29	ENSURE PROPER ASSOCIATION OF WORDS	TIE WORDS TO INTENDED SUBJECT	SA "SQUIGGLE"	label (TOUCH REFERENT)	A32
30	UNIQUE SHEET REFERENCE	CHRONOLOGICAL CREATION	SA C-NUMBER	AUTHOR INITS INTEGER	A41
31	UNIQUE BOX REFERENCE	PATH DOWN TREE FROM BOX NUMBERS	SA NODE NUMBER (BOX NUMBERS)	A, D, OR M ≤ PARENT # ≤ BOX #	A42
32	SAME FOR MULTI-MODELS	PRECEDE BY MODEL NAME	SA MODEL NAME	MODEL NAME/NODE#	A42
33	UNIQUE INTERFACE REFERENCE	ICOM WITH BOX NUMBER	SA BOX ICOM	BOX# ≤ ICOM CODE	A43
34	UNIQUE ARROW REFERENCE	FROM - TO	PAIR OF BOX ICOMs	BOX $ICOM_1$ BOX $ICOM_2$	A44
35	SHOW CONTEXT REFERENCE	SPECIFY A REFERENCE POINT	SA REF.EXP. "DOT"	A122.411 "WHICH SEE"	A45
36	ASSIST CORRECT INTERPRETATION	SHOW DOMINANCE GEOMETRICALLY (ASSIST PARSE)	STAIRCASE LAYOUT	DOMINANCE	A5
37	ASSIST UNDERSTANDING	PROSE SUMMARY OF MESSAGE	SA TEXT	NODE# ≤ T ≤ INTEGER	A5
38	HIGHLIGHT FEATURES	SPECIAL EFFECTS FOR EXPOSITION ONLY	SA FEOs	NODE# ≤ F ≤ INTEGER	A5
39	DEFINE TERMS	GLOSSARY WITH WORDS & PICTURES	SA GLOSSARY	MODEL NAME ≤ G ≤ INTEGER	A5
40	ORGANIZE PAGES	PROVIDE TABLE OF CONTENTS	SA NODE INDEX	NODE# ORDER	A5

Fig. 2. SA language features.

tions of SA language. As we shall see, however, normally these capabilities for richness are purposely suppressed, for simplicity and immediacy of understanding normally require brevity and conciseness.

Fig. 2 has introduced our subject and has served to point out the precise way in which SA absorbs other languages, but this mode of discourse would make a long and rambling story. I therefore proceed to use SA itself to communicate the intended understanding of Fig. 2. This will not, however, be a perfect, or even a good example of SA communication in action, for the intent of this paper is to guide the reader to an understanding of SA, not to teach how to fully exploit SA diagrams and modeling. The SA diagrams presented here only as figures are incomplete and exhibit both good *and* bad examples of SA expressiveness, as well as showing all the language constructs. Our subject is too complex to treat in a small model, but the figures at least present the reader with some measure of the flexibility of the language.

The reader is forewarned that there is more information in the diagrams than is actually referenced here in text which uses them as "figures." After the paper has been read, the total model can be studied for review and for additional understanding. Everything said here about the SA language and notations applies to each diagram, and most features are illustrated more than once, frequently before they are described in the text. Therefore, on first reading, please ignore any features and notations not explicitly referenced. Non-SA "first-reading" aids are isolated by a bold outline, in the diagrams.

In practical use of SofTech's SADT, a "reader/author cycle" is rigorously adhered to in which (similar to the code-reading phase of egoless structured programming) authors, experts, and management-level personnel read and critique the efforts of each individual SADT author to achieve a fully-acceptable quality of result. (It is in fact this rigorous adherence to quality control which enables production SA models to be relied upon. So far as possible *everything* worthy has been done to make sure that *everything* worthy has been expressed to the level required by the intended readership.)

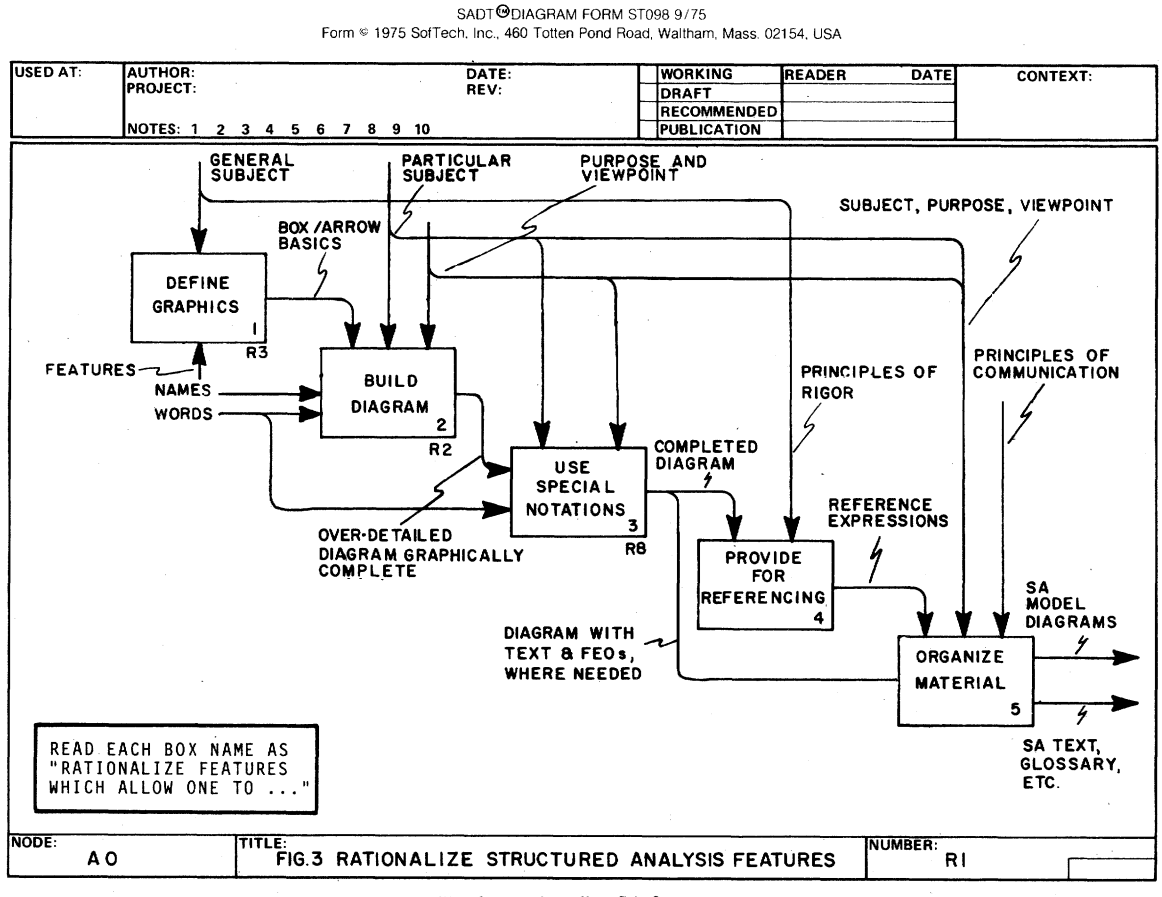

Fig. 3. Rationalize SA features.

X. Purpose and Viewpoint

Fig. 3 is an SA diagram[2] and, by definition, it is a meaningful expression in SA language. It consists of box and arrow graphical notation mixed with words and numbers. Consonant with the tutorial purpose of this paper, I will not, here, try to teach how to *read* a diagram. My tutorial approach aims only to lead to an understanding of what is *in* a diagram.

So we will just begin to examine Fig. 3. Start with the title, "Rationalize structured analysis features"—an adequate match to our understanding of the purpose and viewpoint of this whole paper. We seek to make rational the reasons behind those features. Next read the content of each of the boxes: "Define graphics; build diagram; use special notations; provide for referencing; organize material." These must be the six-or-fewer "parts" into which the titled subject matter is being broken. In this case there are five of them, and sure enough this aspect of SA follows exactly the time-tested outline approach to subject matter. Because our purpose is to have a graphics-based language (like blueprints), once we have decided upon some basics of graphic definition we will use that to build a diagram for some particular subject, adding special notations (presumably to improve clarity), and then because (as with blueprints) we know that a whole collection will be required to convey complex understanding in easy-to-understand pieces, we must provide for a way of referencing the various pieces and organizing the resulting material into what we see as an understandable whole.

Now, I have tried to compose the preceding long sentence about Fig. 3 using natural language constructs which, if not an exact match, are very close to terms which appear directly in Fig. 3. In fact, the reader should be able to find an exact correspondence between things which show in the figure and every important syntactic and semantic point in each part of the last sentence of the preceding paragraph, although the diagram has more to it than the sentence. Please reread that sentence now and check out this correspondence for yourself. In the process you will notice that considerable (though not exhaustive) use is made of information which is not *inside* the boxes, but instead is associated with the word-and-arrow structure of the diagram *outside* the boxes. This begins to show why, although SA in its basic backbone does follow the standard outline pattern of presentation, the box-and-arrow structure conveys a great deal more information than a simple topic outline (only the box *contents*) could possibly convey.

XI. The First Detail View

Fig. 4 is another SA diagram. Simpler in structure than the diagram of Fig. 3, but nonetheless with much the same "look."

[2] The SADT diagram form itself is © 1975, SofTech, Inc., and has various fields used in the practice of SADT methodology.

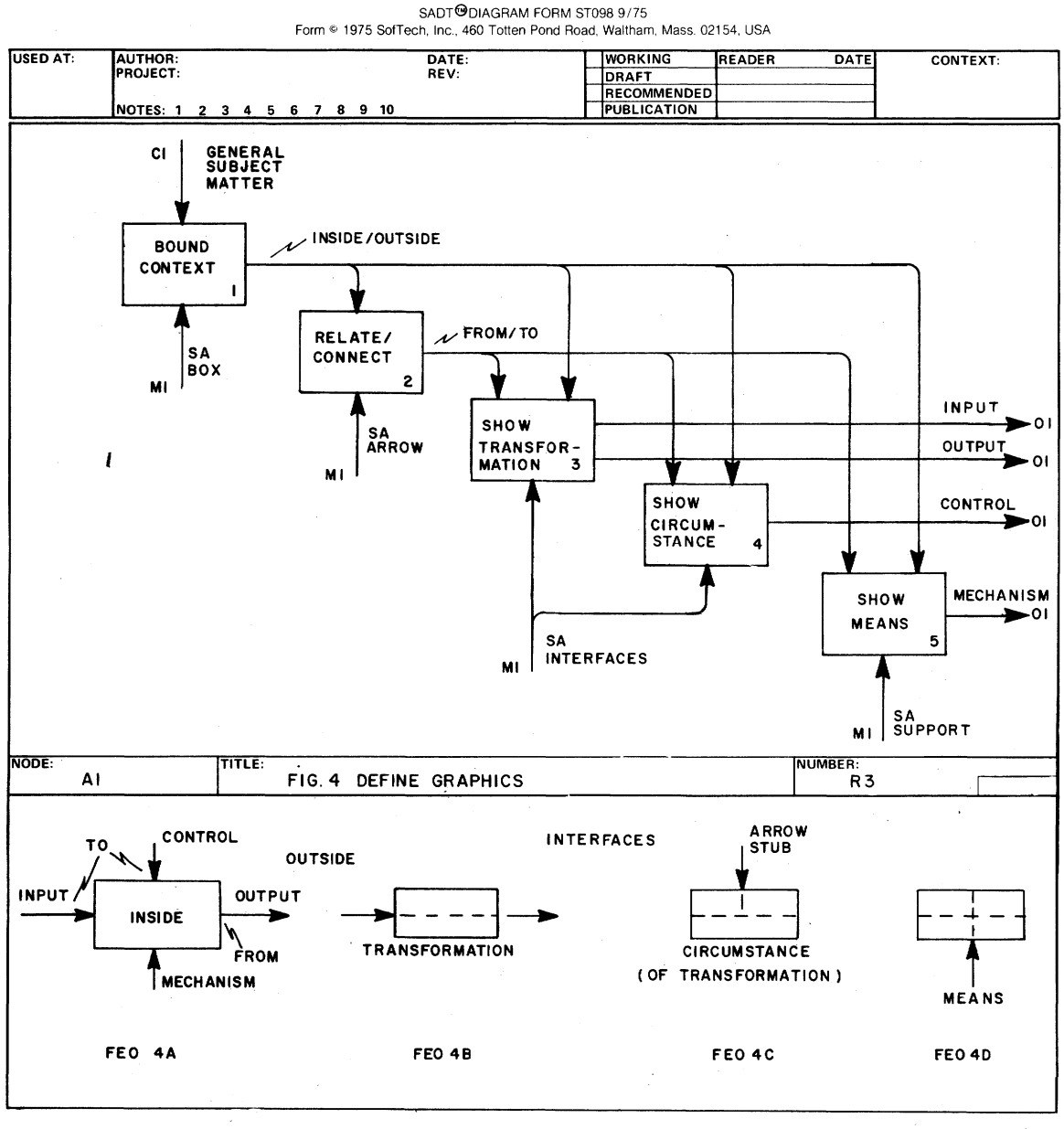

Fig. 4. Define graphics.

The title, "Define graphics," is identical to the name inside the first box of Fig. 3, which is here being broken into five component worthy pieces, called the *nested factors* in SA terminology. Again the words written inside the boxes are legible, but are they understandable? How can "Bound context; relate/connect; show transformation; show circumstance; show means," be considered parts of "Define graphics?" It is not very clear, is it? It would seem that something must be wrong with SA for the touted understandability turns out to be, in fact, quite obscure!

Look at Fig. 4 again and see if you don't agree that we have a problem—and see if you can supply the answer to the problem.

The problem is not with SA at all, but with our too-glib approach to it. SA is a rigorous language and thereby is unforgiving in many ways. In order for the communication of understanding to take place we ourselves must understand and conform to the rules of that rigor. The apparent obscurity should disappear in a twinkling once the following factor is pointed out: namely, *always be sure to do your understanding in the proper context*. In this case, the proper context was established by the title of Fig. 3, "Rationalize structured analysis features," and the purpose, to define graphical concepts and notations for the purpose of representing and conveying understanding of subject matters. Now, if we have all of that firmly implanted in our mind, then surely the name in Box 1 of Fig. 4 should be amply clear. Read, now, Box 4. 1[3] for yourself, and see if that clarity and communication of intended understanding does not take place.

[3]To shorten references to figures, "Box 4. 1" will mean "Box 1 of Fig. 4," etc. in the following discussion.

You see, according to the diagram, the first feature of defining graphics is to "Bound the context"—precisely the subject we have just been discussing and precisely the source of the apparent obscurity which made SA initially appear to be on shaky ground. To aid first reading of the figures, a suggested paraphrasing of the intended context is given in a bold box on each of the other diagrams (see Fig. 3).

As we can see from the section of Fig. 4 labeled FEO 4A[4] the general subject matter is isolated from the rest of all subject matter by means of the SA box which has an inside and an outside (look at the box). The only thing we are supposed to consider is the portion of that subject matter which is *inside* the box—so the boundary of the box does bound the context in which we are to consider the subject.

XII. The SA Box as Building Block

We lack the background (at this point) to continue an actual reading of Fig. 4, because it itself defines the basic graphic notations used in it. Instead, consider only the sequence of illustrations (4A-4D) labeled FEO. FEO 4A shows that the fundamental building block of SA language notation is a box with four sides called INPUT, CONTROL, OUTPUT, and MECHANISM. As we have seen above, the bounded piece of subject matter is *inside* the box, and, as we will see, the actual boundary of the box is made by the collection of *arrow stubs* entering and leaving the box. The bounded pieces are related and connected (Box 4.2) by SA arrows which go *from* an OUTPUT of one box *to* the INPUT or CONTROL of another box, i.e., such arrow connections make the *interfaces* between subjects. The names INPUT and OUTPUT are chosen to convey the idea that (see FEO 4B and Box 4.3) the box represents a *transformation* from a "before" to an "after" state of affairs. The CONTROL interface (see FEO 4C and Box 4.4) interacts and constrains the transformation to ensure that it applies only under the appropriate circumstances. The combination of INPUT, OUTPUT, and CONTROL fully specifies the bounded piece of subject, and the interfaces relate it to the other pieces. Finally, the MECHANISM *support* (not interface, see FEO 4D and Box 4.5) provide means for realizing the complete piece represented by the box.

We will see shortly why Fig. 4 contains no INPUT arrows at all, but except for that anomaly, this description should make Fig. 4 itself reasonably understandable. (Remember the context—"Rationalize the features of SA language which allow one to define graphic notation for....") The diagram (with FEO's and discussion) is the desired rationalization. It fits quite well with the idea of following the maxim. We don't mind breaking *everything* about a bounded piece of subject matter into pieces as long as we are sure we can express completely how all those pieces go back together to constitute the whole. Input, output, control, and mechanism provide that capability. As long as the right mechanism is provided, and the right control is applied, whatever is inside the box can be a valid transformation of input to output. We now must see how to use the "foreign" language names and labels of boxes and arrows. Then we can start putting SA to work.

[4]This notation refers to the sequence of imbedded illustrations in Fig. 4 which are "For exposition only" (FEO).

XIII. Using the Basics for Understanding

Fig. 4, and especially FEO 4A, now that we have digested the meaning of the diagram itself, has presented the basic box-and-arrow-stubs-making-useful-interfaces-for-a-bounded-piece-of-subject-matter building block of SA. We now can start to use the input, output, control, and mechanism concepts to further our understanding. Knowing even this much, the power of expression of SA diagrams beyond that of simple outlining will start to become evident.

Fig. 5, entitled "Build diagram," details Box 3.2. Referring back to Fig. 3 and recalling the opening discussion of its meaning (which we should do in order to establish in our mind the proper context for reading Fig. 5) we recall that the story line of Fig. 3 said that after Box 3.1 had defined the arrow and box basics, then we would build an actual diagram with words and names for a particular subject in accordance with a purpose and viewpoint chosen to convey the appropriate understanding. Looking at Box 3.2 in the light of what we have just learned about the box/arrow basics in Fig. 4, we can see that indeed the inputs are words and names, which will be transformed into a diagram (an over-detailed, but graphically complete diagram, evidently). Even though the mechanism is not specified, it is shown that this diagramming process will be controlled by (i.e., constrained by) the graphic conventions, subject, and viewpoint. Now refer to Fig. 5 with this established context and consider its three boxes:—"Build box structure; build arrow structure; build diagram structure." That matches our understanding that a diagram is a structure of boxes and arrows (with appropriate names and labels, of course). Study Fig. 5 yourself briefly keeping in mind the points we have discussed so far. You should find little difficulty, and you will find that a number of the technical terms that were pure jargon in the tabulated form in Fig. 2 now start to take on some useful meaning. (Remember to ignore terms such as "ICOM" and "DRE," to be described later.)

If you have taken a moment or so to study Fig. 5 on your own, you probably have the impression things are working all right, but you are still not really sure that you are acquiring the intended level of understanding of Fig. 5. It seems to have too many loose ends semantically, even though it makes partial sense. If this is your reaction, you are quite right. For more detail and information is needed to make all the words and relationships take on full meaning. Fig 5 does indeed tell *everything* about "Build diagram" in its three boxes, which are themselves reasonably understandable. But we need more information for many of the labels to really snap into place. This we will find in the further detailing of the three boxes. Context *orients* for understanding (*only* orients!); details *enable* understanding (and strengthen context).

Fig. 6 provides the detailing for Box 5.1. Especially for this diagram, it is important to keep in mind the appropriate context for reading. It is not "*Draw* an SA diagram," but to motivate the *features* of SA. Thus, when we read the title, "Build

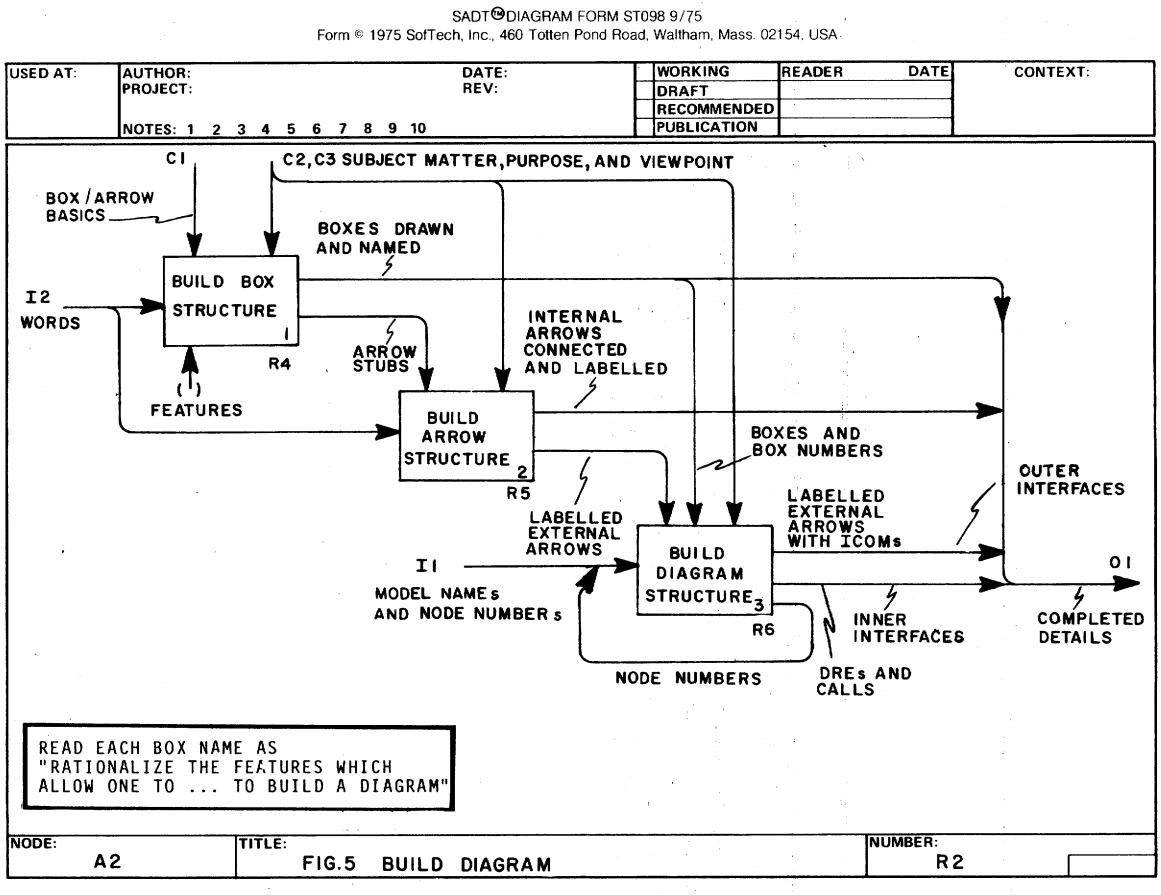

Fig. 5. Build diagram.

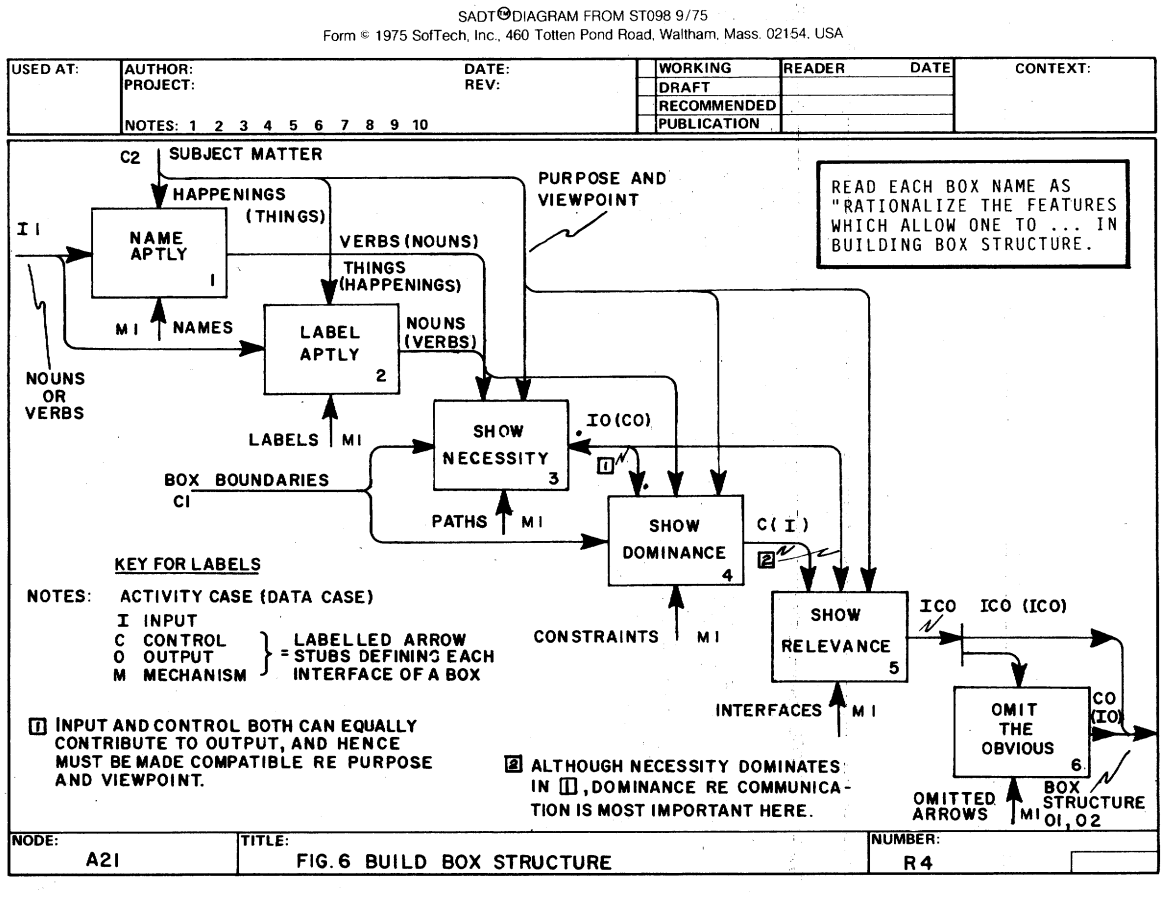

Fig. 6. Build box structure.

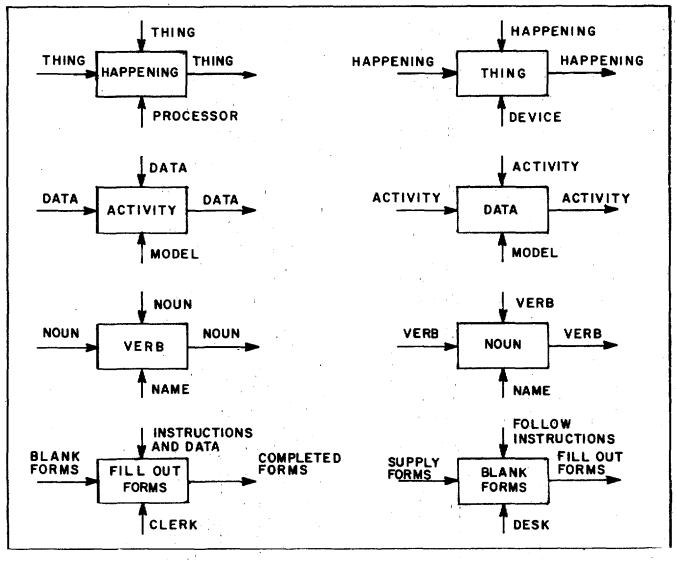

Fig. 7. Duality of activities and data.

box structure," of Fig. 6, we must keep in mind that the worthy piece of subject matter is not *how* to build box structure, nor even the features which *create* box structure, but motivation for *explanation* of the features which allow box structure to represent the bounded context subject matter. This actually is a very sophisticated subject and normally we would only be diagramming it after we had already prepared rather complete models of the "how to" of SA so that many of the terms and ideas would already be familiar. In this paper, however, the opening discussion must serve instead. The next four sections discuss Fig. 6.

XIV. Duality of Data and Activities

Recall that a complete SA model has to consider both the things and happenings of the subject being modeled. Happenings are represented by an activity decomposition consisting of activity diagrams and things are represented by data decomposition consisting of data diagrams. The neat thing about SA language is that both of these complementary but radically different aspects are diagrammed using exactly the same four-sided box notation. Fig. 7 illustrates this fact. The happening/activity and thing/data domains are completely dual in SA. (Think of an INPUT activity on a data box as one that creates the data thing, and of OUTPUT as one that uses or references it.) Notice that mechanism is different in interpretation, but the role is the same. For a happening it is the *processor*, machine, computer, person, etc., which makes the happening happen. For a thing it is the *device*, for storage, representation, implementation, etc. (of the thing).

A quick check of Fig. 4 shows that mechanism's purpose is to show the means of realization, and that it is *not an interface* but is instead something called "support" in SA (described later in Section XXIII). For either activity or data modeling, a support mechanism is *a complete model*, with both data and activity aspects. As Fig. 7 shows, that complete "real thing" is *known by its name*, whereas things and happenings are identified by nouns and verbs (really nominal expressions and verbial expressions). With this in mind, we can see that the first two boxes in Fig. 6 motivate the naming and labeling features of SA to do or permit what Fig. 7 requires–boxes are named, and arrows are labeled, with either nouns or verbs as appropriate to the aspect of the model, and of course, in accordance with the intended purpose and viewpoint of the subject matter.

XV. Constraints

We will consider next Boxes 6.3 and 6.4 together, and with some care, for this is one of the more subtle aspects of SA–the concept of a *constraint*–the key to well structured thought and well structured diagrams. The word *constraint* conjures up visions of opposing forces at play. *Something can be constrained only if there is something stronger upon which the constraining force can be based.* It might seem that from the ideas of SA presented so far, that that strong base will be provided by the rigorously defined bounded context of a box. Given a strong boundary, it is easy to envision forces saying either to stay inside the boundary or to stay outside the boundary. It is a pleasing thought indeed, and would certainly make strong structure in both our thinking and our diagrams. The only trouble is it does not work that way (or at least not immediately), but in fact it is just the opposite! In SA thinking *it is the constraints that make the boundaries, not the other way around.* This is a tricky point so we will approach it slowly. It is still true that a constraint, to be a constraint, has to have something to push against. If it is not the bounded-context boundary, then what is it?

The subtle answer is that *the purpose and viewpoint of the model provide the basis for all constraints* which in turn provide the strength and rigidity for all the boundaries which in turn create the inescapable structure which forces correct understanding to be communicated. This comes about through the concepts of *necessity* and *dominance*, which are the subjects of Boxes 6.3 and 6.4. Dominance sounds much like constraint, and we will not be surprised to find it being the purveyor of constraint. But "necessity" has its own subtle twist in this, the very heart of SA. Therefore we must approach it, too, with some deliberation.

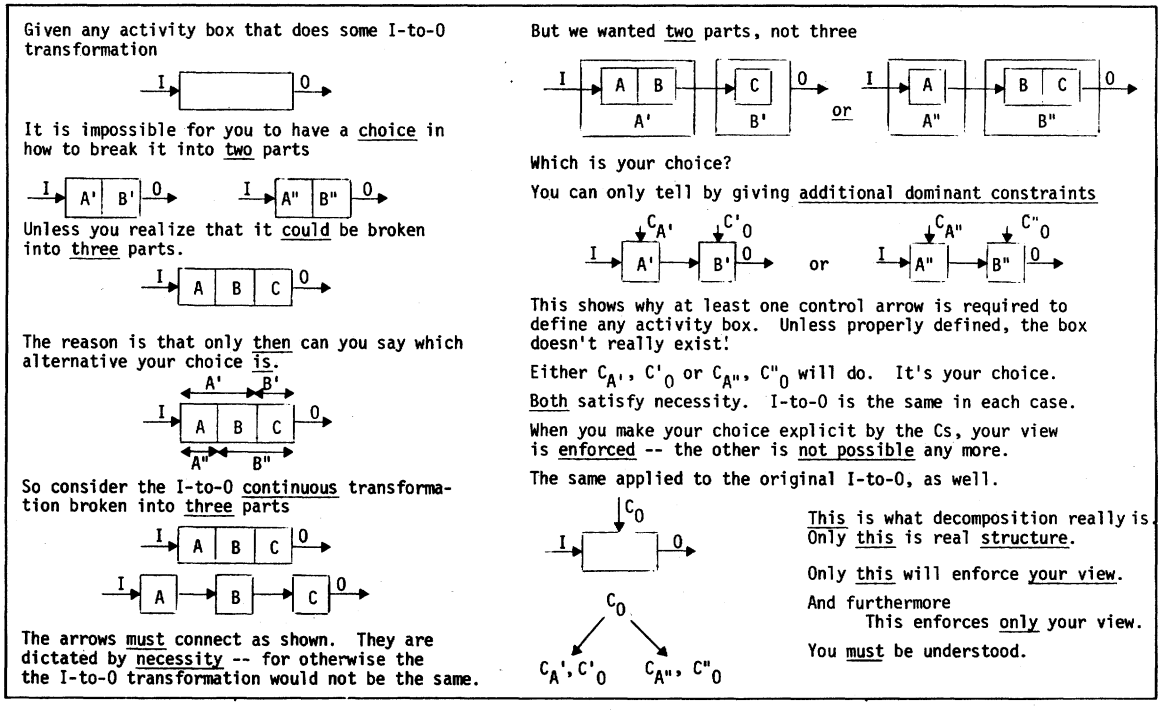

Fig. 8. Dominance and necessity.

Fig. 8 tells the story in concise form. Please read it now. Then please read it again, for all experience with SA shows that this simple argument seems to be very subtle and difficult for most people to grasp correctly. The reason is that the *everything* of the SA maxim makes the I-to-O *necessity* chain the *weakest possible structure*—akin to *no structure at all*. It merely states a fact that *must be true* for every SA box, because of the maxim. Therefore *dominant constraints*, expressed by the control arrows for activity boxes are, in fact, the *only* way possible, to impose structure. Furthermore, that enforcement of structure is unique and compelling—no other structure can be (mis-) understood in place of that intended by the SA author. This is, of course, all mediated by the effectiveness with which the SA author wields the chosen non-SA language used for names of boxes and labels on arrows, but the argument presented in this paper holds, nonetheless. This is because whenever the imprecision of the non-SA language intrudes, more SA modeling (perhaps even with new purpose and viewpoint for greater refinement, still, of objectives) is forced by the reader/author cycle of the SADT discipline.

XVI. The Rule of Omission

Now consider Boxes 6.5 and 6.6 "Show relevance" and "Omit the obvious." These two ideas follow right along with the above discussion. Namely, in the case of activity diagramming, *if inputs are relevant* (i.e., if they make a strong contribution to understandability) *then they are drawn*. But on the other hand, since the important thing is the structure imposed by the control dominance and output necessity, and inputs *must* be supplied in any case for those outputs to result, *obvious inputs can and should be omitted from* the box structure of SA *activity diagramming*. In other words, whenever an obvious input is omitted in an activity diagram, the reader knows that (because of the SA maxim) whatever is needed will be supplied in order that the control and output which *are* drawn can happen correctly. Omitting the obvious makes the understandability and meaning of the diagram much stronger, because inputs when they *are* drawn are known to be important and nonobvious. Remember that SA diagrams are not wiring diagrams, they are vehicles for communicating understanding.

Although activity and data are dual in SA and use the same four-sided box notation, they are not quite the same, for the concept of dominance and constraint in the *data* aspect centers on *input* rather than control. In the data case, the weak chain of necessity is C-O-C-O-· · · not I-O-I-O-· · · as it is in the activity case. The reason comes from a deep difference between space and time (i.e., between things and happenings). In the case of the happening, the dominant feature is the control which says when to cut the transformation to yield a desired intermediate result, because the "freedom" of happenings is in time. In the case of things, however, the "freedom" which must be constrained and dominated concerns which version of the thing (i.e., which part of the data box) is the one that is to exist, regardless of time. The input activity for a data box "creates" that thing in that form and therefore it is the dominant constraint to be specified for a data box. An unimportant control activity will happen whenever needed, and may be omitted from the diagram.

Therefore, the rule regarding the obvious in SA is that *controls may never be omitted from activity diagrams and inputs may never be omitted from data diagrams*. Fig. 6 summarizes all of this discussion.

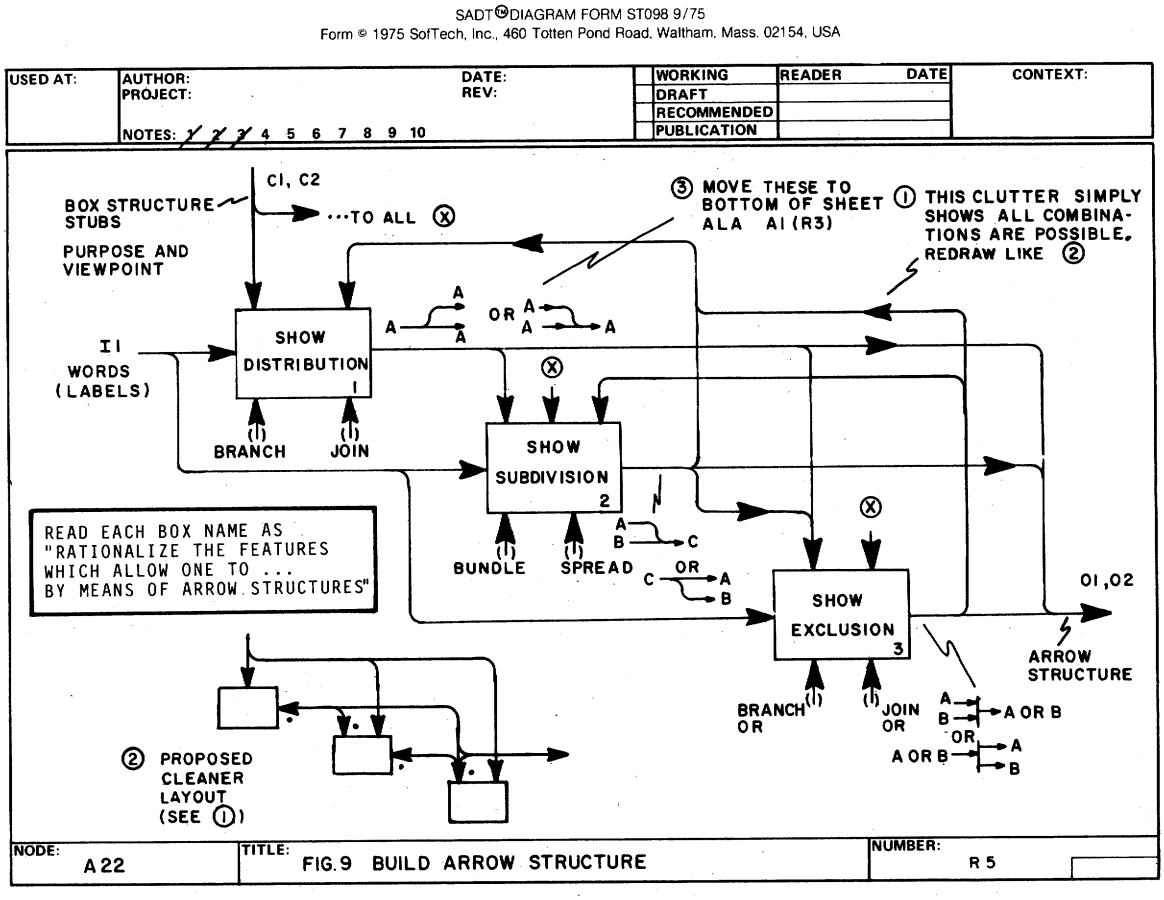

Fig. 9. Build arrow structure.

XVII. Strengthening of Boundaries

Recall that a constraint does need something to push on. What is that? The answer is the one originally proposed, but rejected. *Constraints are based on the boundary of a bounded context–but of the single box of the parent diagram which the current diagram details.* (A parenthesized hint that this would turn out to be the case *does* appear in the original rejection of the view that boundaries provide the base for constraints, above.) In other words, the constraints represented by the arrows on a diagram are *not* formed by the several boundaries of the boxes of *that* diagram, but all are based on the single boundary of the corresponding box in the *parent* diagram. As was stated above in Section XV, the constraints form the box boundary interfaces and define the boxes, not the other way around. The strength of those constraints comes from the corresponding "push" passed through from the parent. As the last "C_O" portion of Fig. 8 indicates, this hierarchic cascading of constraints is based entirely on the *purpose* of the model as a whole (further constrained in spread by the limited *viewpoint* of the model) as it is successively decomposed in the hierarchic layering forced by the six-or-fewer rule of the SA maxim.

All of this is a direct consequence of the *everything* of the SA maxim, and may be inferred by considering Fig. 8 recursively. The boundary of the top-most box of the analysis is determined entirely by the subject matter, purpose, and viewpoint of the agreed-upon outset understanding. ("Tell 'em whatcha gonna tell 'em.") Then each of the subsequent constraints derives its footing only insofar as it continues to reflect that subject, purpose, and viewpoint. And, each, in turn, provides the same basis for the next subdivision, etc. Inconsistencies in an original high-level interpretation are ironed out and are replaced by greater and greater precision of specific meaning.

Even though the basis for all of the constraint structure is the (perhaps ill-conceived, ambiguous, ill-defined) *outer* boundary, that boundary and the *innermost* boundary, composed of the collective class of all the boundaries of the finest subdivision taken together, are merely two representations of the *same* boundary–so that *strengthening of the inner boundary through extensive decomposition automatically strengthens the outer boundary*. It is as though the structured analyst (and each of his readers as well) were saying continually "My outermost understanding of the problem as a whole can only make sense, now that I see all this detail, if I refine my interpretation of it in this, this, and this precise way." This is the hidden power of SA at work. This is how SA greatly amplifies the precision and understandability of any natural or formal language whose nouns and verbs are imbedded in its box and arrow structure.

XVIII. Arrow Connections

Fig. 6, which led to this discussion, detailed Box 5.1 "Build box structure"; Fig. 9 decomposes Box 5.2, "Build arrow

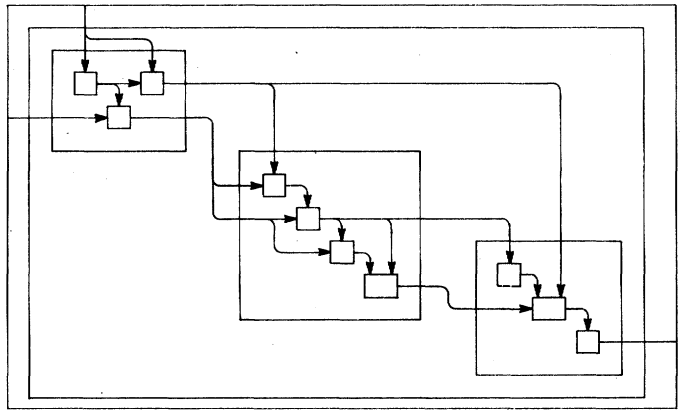

Fig. 10. Nested factors.

structure." From Fig. 5 we see that the controlling constraints that dominate Box 5.2 are the subject matter, purpose, and viewpoint, as we would expect, along with the "arrow stubs" which resulted from building each box separately. The outputs are to be internal arrows connected and labeled, as well as labeled external arrows. A relevant (i.e., nonobvious) input is the collection of words—nouns or verbs—for making those labels.

With this context in mind, we are now ready to look at Fig. 9, "Build arrow structure." Here is an example of the use of non-English language to label arrows. Small graphical phrases show the intended meaning of "branch" and "join" for distribution, and "bundle" and "spread" with respect to subdivision, as well as two forms of logical OR for exclusion. We have seen many examples of these in use in the diagrams already considered, so that the ideas should be quite transparent.

The little pictures as labels show how the labels attach to arrows to convey the appropriate meaning. In most good SA diagramming the OR's are used very sparingly—only when they materially assist understanding. In most circumstances, the fact that arrows represent constraints either of dominance or necessity supplies the required understanding in clearer form merely by topological connection. This also is the reason why there is no graphical provision for the other logical functions such as AND, for they are really out of place at the level of communication of basic SA language. In order for them to have an appropriate role, the total context of interpretation of an SA model must have been drawn down very precisely to some mathematical or logical domain at which point a language more appropriate to that domain should be chosen. Then logical terms in the nominal and verbial expressions in labels can convey the conditions. This is preferable to distorting the SA language into a detailed communication role it was not designed or intended to fulfill.

XIX. Boundaries

Fig. 12, "Build diagram structure," will provide detailing for the third and last box of Fig. 5. It is needed as a separate consideration of this motivation model because the building of box structure (Box 5.1, detailed in Fig. 6) and arrow structure (Box 5.2, detailed in Fig. 9) only cover arrows between boxes in a single diagram—the *internal* arrows. Box 4.2 requires that *every* arrow which relates or connects bounded contexts must participate in both a *from* and a *to* interface. Every *external* arrow (shown as the second output of Box 5.2) will be missing either its source (from) or its destination (to) because the relevant boxes do not appear on this diagram. As the relationship between Boxes 5.2 and 5.3 in Fig. 5 shows, these labeled arrows are indeed a dominant constraint controlling Box 3, "Build diagram structure."

Fig. 10 helps to explain the story. This is a partial view of three levels of nesting of SA boxes, one within the other, in some model (not an SA diagram). Except for three arrows, every arrow drawn is a complete from/to connection. The middle, second-level box has four fine-level boxes within it, and it in turn is contained within the largest box drawn in the figure. If we consider the arrows in the middle, second-level box, we note that only two of them are internal arrows, all of the others being external. But notice also that every one of those external arrows (with respect to that middle-level box) are in fact *internal* with respect to the model as a whole. Each of those arrows does go from one box to another box—a lowest-level box in each case. In completing the connection, *the arrows penetrate the boundaries of the middle-level boxes* as though those boundaries were not there at all. In fact, there are only two real boundaries in all of Fig. 10—the two boundaries characteristic of every SA decomposition. These are 1) the *outer boundary* which is the outermost edge of Fig. 10, itself, and 2) the *inner boundary* which is the entire set of edges of all of the lowest-level boxes drawn in Fig. 10, considered as a single boundary. As was stated above, the SA maxim requires that the outer boundary and the inner boundary must be understood to be *exactly the same* so that the subject is merely decomposed, not altered in any way.

XX. Parents and Children

To understand how the structuring of Fig. 10 is expressed in SA terms we must be clear about the relationship between boundaries and interfaces, boxes and diagrams, and the parent/child relationship. Fig. 11 lays all of this out. In the upper right appears the diagram for the largest box drawn in Fig. 10, and in the lower left appears the diagram for the central middle-level box which we were discussing. The first thing to notice is that the diagrams are here drawn as though they were

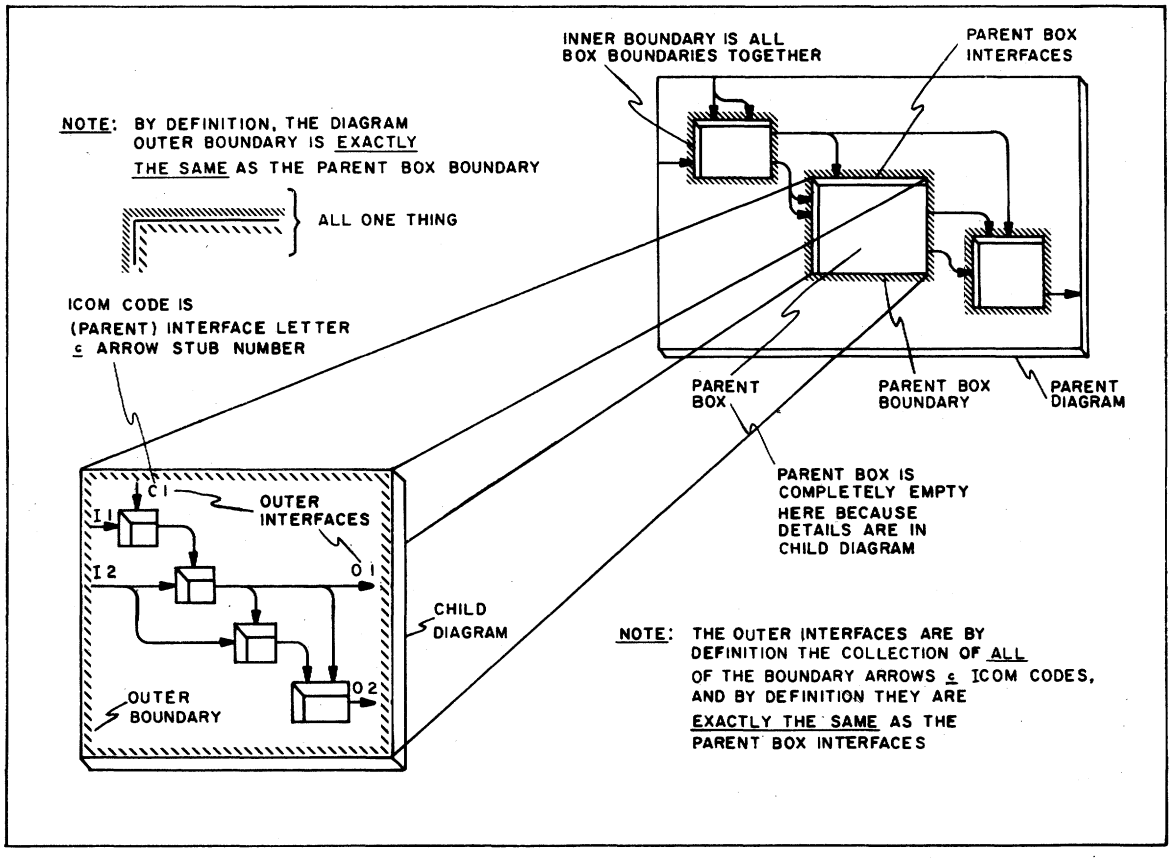

Fig. 11. Boundaries and interfaces.

punched out of Fig. 10, (like cutting cookies from a sheet of cookie dough). Although the dimensions are distorted, the note in the upper left points out that, by definition, *the diagram outer boundary is actually the same as the parent box boundary* (i. e., the current child diagram is the "cookie" removed from the sheet of dough and placed to one side).

Fig. 11 also points out that, just as for the hierarchic decomposition as a whole, the *inner boundary of the parent diagram is the collection of all its child box boundaries considered as a single entity*. Notice the terminology—with respect to the current child diagram, one of the boxes in the parent diagram is called the *parent box* of the child diagram. By definition of Fig. 4, that *parent box boundary* is the collection of *parent box interfaces and support* which compose it. Since we have just established that the outer boundary of the current diagram is the same as the corresponding parent box boundary, the parent box edges (interfaces *or* support) which compose the parent box boundary must somehow match the outer edges of the child diagram. This is the connection which we seek to establish rigorously.

By Fig. 10 we know that the external arrows of the child diagram penetrate through the outer boundary and are, in fact, the *same* arrows as are the stubs of the interfaces and support which compose the parent box boundary. Therefore, the connection which has to be made is clear from the definition. But for flexibility of graphic representation, *the external arrows of the current diagram need not have the same geometric layout, relationship, or labeling* as the corresponding stubs on the parent diagram, which are drawn on a completely different (parent) diagram.

In order to allow this flexibility, we construct a special code-naming scheme called *ICOM codes* as follows: An ICOM code begins with one of the letters I-C-O-M (standing for INPUT, CONTROL, OUTPUT, MECHANISM) concatenated with an integer which is obtained by considering that the stubs of the corresponding parent box edge are numbered consecutively from top to bottom or from left to right, as the case may be. With the corresponding ICOM code written beside the unattached end of the arrow in the current diagram, that arrow is no longer called "external," but is called a *boundary arrow*. Then *the four outer edges of the child diagram are, by definition, the four collections of ICOM boundary arrows* which are, by definition, exactly the same as the corresponding parent box edges, as defined by Fig. 4 and shown in Fig. 11. Thus even though the geometric layout may be radically different, the rigor of interconnection of child and parent diagrams is complete, and the arrows are continuous and unbroken, as required. Every diagram in this paper has ICOM codes properly assigned.

The above presentation is summarized in the first two boxes of Fig. 12 and should be clear without further discussion. Boxes 12.3 and 12.4 concern the SA language notations for establishing the relationships between the child and parent diagram cookies by means of a *detail reference expression* (DRE) or an SA *call*. We will consider these shortly. For now it is sufficient to note that the topics we have considered here

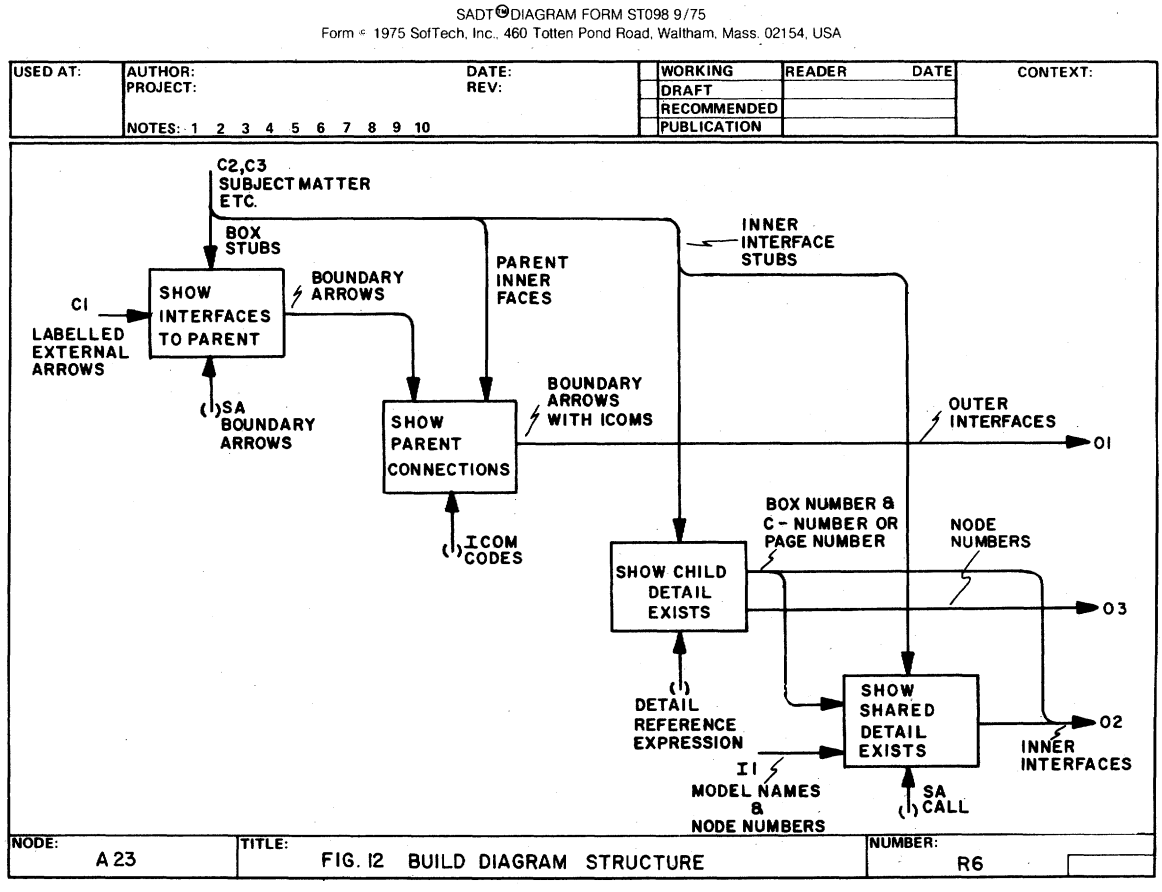

Fig. 12. Build diagram structure.

complete the detailing of Fig. 5, "Build diagram"—how the box structure and arrow structure for individual diagrams are built, and then how the whole collection of diagrams is linked together in a single whole so that *everything* of the top-most cookie (treated as a cookie sheet from which other cookies are cut with zero width cuts) is completely understandable. Each individual diagram itself is only a portion of the cookie dough with an outer boundary and an inner boundary formed by the decomposition operation. Nothing is either gained or lost in the process—so that the SA maxim is rigorously realized. Everything can indeed be covered for the stated purpose and viewpoint. We now complete our presentation of the remaining items in the 40 features of Fig. 2, which exploit further refinements of notation and provide orderly organization for the mass of information in a complete SA model.

XXI. Word Notes

In SA language, not everything is said in graphical terms. Both words and pictures are also used. If the diagram construction notations we have considered so far were to be used exclusively and exhaustively, very cluttered and nonunderstandable diagrams would result. Therefore SA language includes further simplifying graphic notations (which also increase the expressive power of the language), as well as allowing nongraphic additional information to be incorporated into SA diagrams. This is the function of Fig. 13 which details Box 3.3. Fig. 13 points out that the (potentially) cluttered diagram is only graphically complete so that special *word* notations are needed. Furthermore, special arrow notations can supply more clarity, to result in a complete *and* understandable diagram.

We will not further detail Box 13.2. We merely point out that its output consists of three forms of verbal additions to the diagram. The first two—NOTES and [n] footnotes—are actual parts of the diagram. The diagrams we have been examining have examples of each. The third category, (n) metanotes, are *not* parts of the diagram themselves, but are instead notes *about* the diagram. The (n) metanotes have only an observational or referential relation to the actual information content of these diagrams and therefore they do not in any way alter or affect the actual representational function of the SA language, either graphical or verbal. There is no way that information in (n) metanotes can participate in the information content of the diagrams, and therefore they should not be used in an attempt to affect the interpretation of the diagrams themselves, but only for mechanical operations regarding the diagram's physical format or expression. Examples are comments from a reader to an author of a diagram suggesting an improved layout for greater understandability. A few examples are included on the diagrams in this paper. The [n] footnotes are used exclusively for allowing large verbal expressions to be concisely located with respect to tight geometric layout, in addition to the normal footnoting function commonly found in textual information.

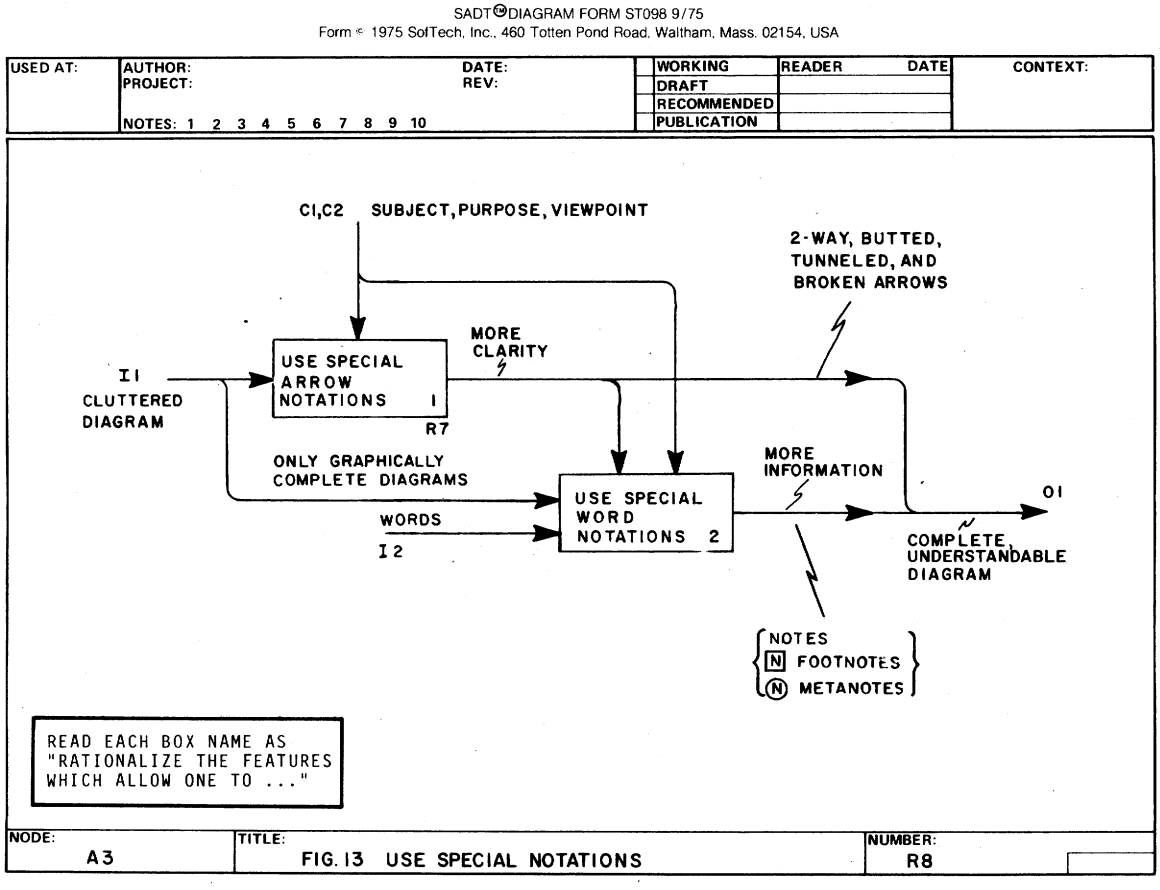

Fig. 13. Use special notations.

XXII. Special Graphic Notations

Fig. 14 provides the motivation for four very simple additions to the graphic notation to improve the understandability of diagrams. With respect to a specific aspect of the subject matter, two boxes sometimes really act as one box inasmuch as each of them shares one portion of a well-defined aspect of the subject matter. In this case, arrowheads with dots above, below, or to the right of the arrowhead are added instead of drawing two separate arrows as shown in FEO 14A. Two-way arrows are a form of bundling, however, *not* a mere shorthand notation for the two separate arrows. If the subject matters represented by the two separate arrows are not sufficiently similar, they should *not* be bundled into a two-way arrow, but should be drawn separately. Many times, however, the two-way arrow *is* the appropriate semantics for the relationship between two boxes. Notice that if other considerations of diagrams are sufficiently strong, the awkward, nonstandard two-way arrow notations shown also may be used to still indicate dominance in the two-way interaction.

SA arrows should always be thought of as conduits or pipelines containing multistranded cables, each strand of which is another pipeline. Then the branching and joining is like the cabling of a telephone exchange, including trunk lines. Box 14.2 is related to both two-way arrows and pipelines, and points out that a one-way pipeline stub at the parent level may be shown as a two-way boundary arrow in the child. This is appropriate since, with respect to the communication of understanding at the parent level, the relationship between boxes is one-way, whereas when details are examined, two-way cooperation between the two sets of detailing boxes may be required. An example is the boss-worker relation. The boss (at parent level) provides one-way command, but (at the child diagram level) a two-way interchange between worker and boss may be needed to clarify details.

Box 14.3 motivates an additional and very useful version of Box 6.6 ("Omit the obvious"). In this case, instead of omitting the obvious, we only postpone consideration of necessary detail until the appropriate level is reached. This is done by putting parentheses around the unattached end of an external arrow, or at the interface end of a parent box stub. The notation is intended to convey the image of an arrow "tunneling" out of view as it crosses a parent box inner boundary only to emerge later, some number of levels deeper in a child's outer interface, when that information is actually required. These are known as "tunneled" or "parenthesized" boundary arrows when the sources or destinations are somewhere within the SA model, and as (proper) *external* arrows when the missing source or destination is unspecified (i.e., when the model would need to be imbedded in some context larger than the total model for the appropriate connection to be made).

Finally, Box 14.4 is a seldom-used notation which allows internal arrows themselves to be broken by an ad hoc labeling scheme merely to suppress clutter. Its use is discouraged be-

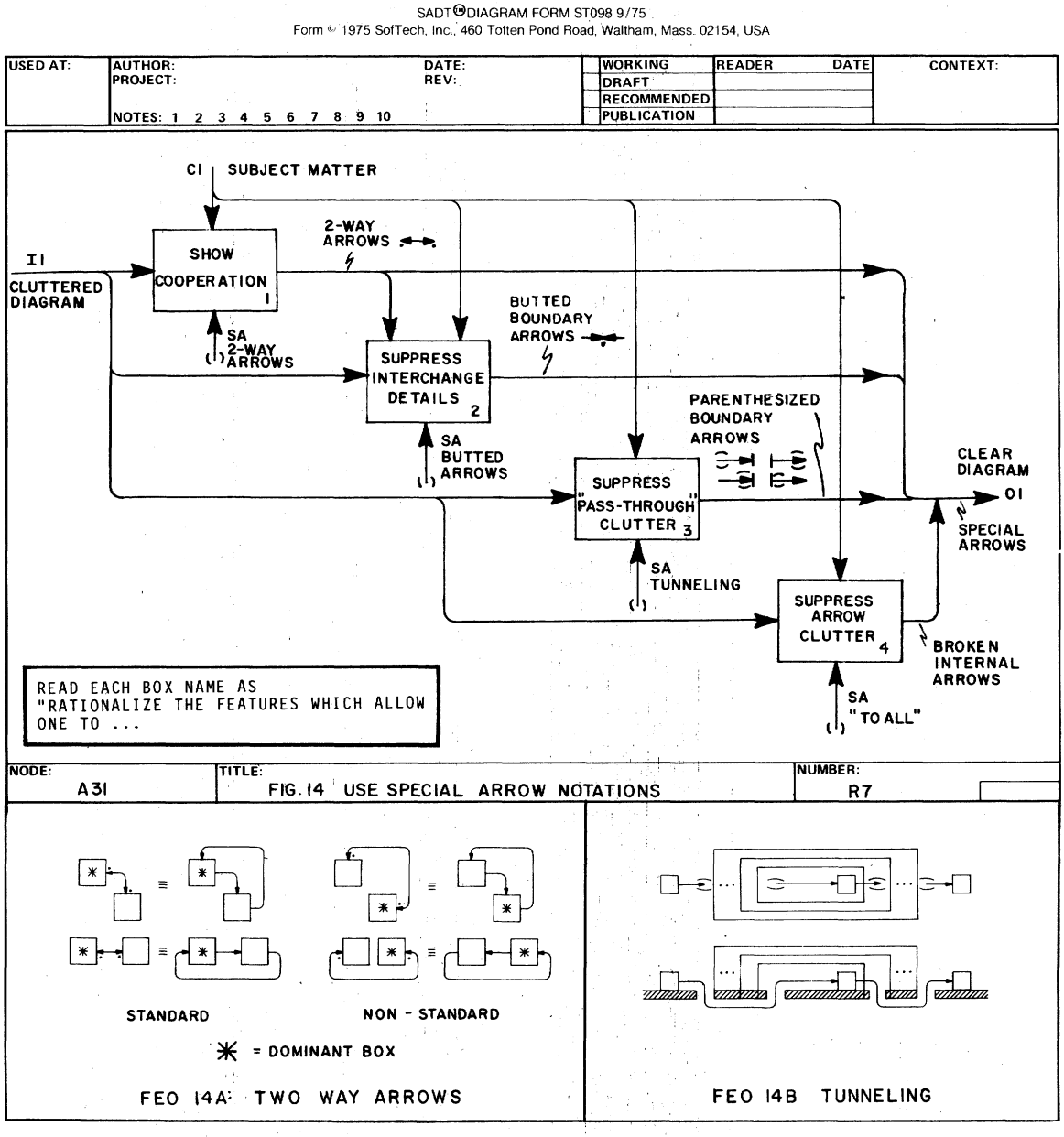

Fig. 14. Use special arrow notations.

cause of its lack of geometric continuity and because its use is forced only by a diagram containing so much information already that it is likely not be be clearly understood and should be redrawn. Examples occur in Fig. 9 just for illustration.

XXIII. The Reference Language

Returning to Fig. 3, we now have considered all of the aspects of basic SA language which go into the creation of diagrams themselves, with the single exception of the detail reference expressions and SA call notation of Fig. 12, which were saved until this point since they relate so closely to Box 3.4, "Provide for referencing."

A complete and unique SA reference language derives very nicely from the hierarchically nested factors imposed by the SA maxim. Diagrams, boxes, interfaces, arrows, and complete contexts can be referenced by a combination of model names, *node numbers* (starting with A for activity or D for data, and derived directly from the box numbers), and ICOM codes. The insertion of a dot, meaning "which see" (i. e. "find the diagram and look at it"), can specify exactly which diagram is to be kept particularly in mind to provide the context for interpreting the SA language. Thus "A122. 4I1" means "in diagram A122, the first input of box 4"; "A1224. I1" means "in diagram A1224, the boundary arrow I4"; "A1224I1" means "the first input interface of node A1224." The SA language rules also allow such reference expressions to degenerate naturally to the minimum needed to be understandable. Thus, for example, the mechanism for showing that the child detailing exists is merely to write the corresponding chronological creation number (called a *C-number*) under the lower right-hand corner of a box on the diagram, as a DRE. (A C-number is the author's intials, concatenated with a sequential integer–

assigned as the first step whenever a new diagram sheet is begun.) When a model is formally published, the corresponding detail reference expression is normally converted into the page number of the appropriate detail diagram. The omission of a detail reference expression indicates that the box is not further detailed in this model. (For all the diagrams considered in this paper, the DRE's have been left in C-number form.)

The *SA call* notation consists of a detail reference expression preceded by a downward pointing arrow stub, and allows sharing of details among diagrams. It will not be covered in this paper beyond the illustration in Fig. 15, which is included here more to illustrate why mechanism support is not an interface (as has repeatedly been pointed out) than to adequately describe the SA call scheme. That will be the subject of a future paper, and is merely cited here for completeness. The SA call mechanism (see also [2]) corresponds very closely to the subroutine call concept of programming languages, and is a key concept in combining multiple purposes and viewpoints into a single model of models.

XXIV. Organizing the Model

The final box of Fig. 3, Box 3.5, "Organize material," is not detailed in this model. Instead, we refer the reader back to the tabulation of Fig. 2 where the corresponding items are listed. In final publication form each diagram is normally accompanied by brief, carefully-structured *SA text* which, according to the reading rules, is intended to be read *after* the diagram itself has been read and understood. The SA text supplements but does not replace the information content of the diagram. Its purpose is to "tell 'em whatcha told 'em" by giving a walk-through through the salient features of the diagram, pointing out, by using the reference language, how the story line may be seen in the diagram. Published models also include glossaries of terms used, and are preceded by a *node index*, which consists of the node-numbered box names in indented form in node number sequence. Fig. 16 is the node index for the model presented in this paper, and normally would be published at the beginning to act as a table of contents.

XXV. Conclusion

The principle of good storytelling style (see Section IV) has been followed repeatedly in this paper. We have provided motivations for each of the 40 SA language features of structured analysis by relating each one to a need for clear and explicit exposition with no loss from an original bounded context. (The "node" column of Fig. 2 maps each feature to a diagram box in the other figures.) In the process, we have seen how the successive levels of refinement strengthen the original statement of purpose and viewpoint, to enforce unambiguous understanding. The best "tell 'em whatcha told 'em" for the paper as a whole is to restudy the SA model in the figures. (Space precludes even sketching the corresponding data decomposition.) The diagrams not only summarize and integrate the ideas covered in the paper, but provide further information, as well.

There are more advanced features of the SA language which will be covered in subsequent papers in the context of applications. In practice, SA turns out to depend heavily on the disciplined thought processes that lead to well-structured analyses expressed in well-structured diagrams. Additional rules and supporting methodology organize the work flow, support the mechanics of the methods, and permit teams of people to work and interact as one mind attacking complex problems. These are covered in SofTech's SADT methodology. The fact that SA incorporates by definition any and all languages within its framework permits a wide variety of natural and artificial languages to be used to accomplish specific goals

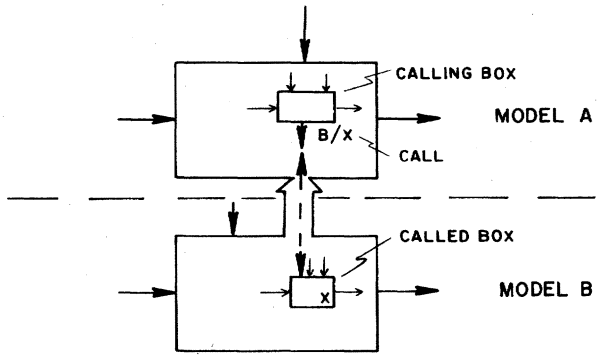

Fig. 15. SA "call" for detailing.

RATIONALIZE SA FEATURES
- A1 DEFINE GRAPHICS
  - A11 Bound Context
  - A12 Relate/Connect
  - A13 Show Transformation
  - A14 Show Circumstance
  - A15 Show Means
- A2 BUILD DIAGRAM
  - A21 Build Box Structure
    - A211 Name Aptly
    - A212 Label Aptly
    - A213 Show Necessity
    - A214 Show Dominance
    - A215 Show Relevance
    - A216 Omit the Obvious
  - A22 Build Arrow Structure
    - A221 Show Distribution
    - A222 Show Subdivision
    - A223 Show Exclusion
  - A23 Build Diagram Structure
    - A231 Show Interfaces to Parent
    - A232 Show Parent Connections
    - A233 Show Child Detail Exists
    - A234 Show Shared Detail Exists
- A3 USE SPECIAL NOTATIONS
  - A31 Use Special Arrow Notations
    - A311 Show Cooperation
    - A312 Supress Interchange Details
    - A313 Supress "Pass-Through" Clutter
    - A314 Supress Arrow Clutter
  - A32 Use Special Word Notations
- A4 PROVIDE FOR (UNIQUE) REFERENCING
  - (A41 Sheet Reference)
  - (A42 Box Reference)
  - (A43 Interface Reference)
  - (A44 Arrow Reference)
  - (A45 Context Reference)
- A5 ORGANIZE MATERIAL

Fig. 16. Node index.

with respect to understanding the requirements for solution. Then those requirements can be translated, in a rigorous, organized, efficient, and, above all, understandable fashion, into actual system design, system implementation, maintenance, and training. These topics also must of necessity appear in later papers, as well as a formal language definition for the ideas unfolded here.

ACKNOWLEDGMENT

The four-sided box notation was originally inspired by the match between Hori's activity cell [3] and my own notions of Plex [4]. I have, of course, benefitted greatly from interaction with my colleagues at SofTech. J. W. Brackett, J. E. Rodriguez, and particularly J. B. Goodenough gave helpful suggestions for this paper, and C. G. Feldmann worked closely with me on early developments. Some of these ideas have earlier been presented at meetings of the IFIP Work Group 2. 3 on Programming Methodology.

REFERENCES

[1] D. T. Ross, "It's time to ask why?" *Software Practise Experience*, vol. 1, pp. 103–104, Jan.–Mar. 1971.
[2] D. T. Ross, J. B. Goodenough, C. A. Irvine, "Software engineering: Process, principles, and goals," *Computer*, pp. 17–27, May 1975.
[3] S. Hori, "Human-directed activity cell model," in *CAM-I*, long-range planning final rep., CAM-I, Inc., 1972.
[4] D. T. Ross, "A generalized technique for symbol manipulation and numerical calculation," *Commun. Ass. Comput. Mach.*, vol. 4, pp. 147–150, Mar. 1961.
[5] D. T. Ross and K. E. Schoman, Jr., "Structured analysis for requirements definition," this issue, pp. 6–15.
[6] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychol. Rev.*, vol. 63, pp. 81–97, Mar. 1956.

Douglas T. Ross, for a photograph and biography, see this issue, p. 5.

Automated Software Engineering Through Structured Data Management

C. A. IRVINE AND JOHN W. BRACKETT

***Abstract*—The Software Engineering Facility (SEF) is a system for software engineering which is specifically designed to support the development of well-engineered software. However, it is not an operating system. Unlike operating systems such as OS/370, EXEC 8, and others, the SEF is not meant to support the execution of applications programs, just as the ordinary operating systems are not intended to specifically support the development of well-engineered applications programs. The SEF, in fact, will run under and use the facilities of such operating systems. It, then, is easily transferable and can be used with various hardware/operating system configurations where it will provide a host-independent software development system. In such a role it will provide to the software developer standard facilities across a variety of host systems.**

The SEF provides the central support for an integrated collection of subsystems, and the subsystems provide appropriate facilities for all phases of the software development process, from requirements definitions through maintenance and enhancement. Each such subsystem contributes in a cohesive way towards the cost-effective development of *well-engineered* software.

The design of the SEF was determined only after a careful analysis of the software development process and a thorough study of recent efforts in the field of software engineering.

***Index Terms*—Requirements definition, software engineering database, software tools, structured programming, top-down design.**

Manuscript received June 14, 1976. This work was supported in part by the Naval Air Development Center, Warminster, PA, under Contract N62269-74-C-0790.

The authors are with SofTech, Inc., Waltham, MA 02154.

I. THE SOFTWARE DEVELOPMENT ENVIRONMENT

THE Software Engineering Facility (SEF) which is discussed in this paper is distinguished by six characteristics, as follows.

1) Its architecture has been profoundly affected by a desire to foster the use of the most effective software engineering principles available.

2) It embodies a unifying design strategy permitting the development of simple tools that are broadly useful.

3) It allows an evolutionary development strategy wherein one develops the most useful tools first and may spread the SEF development costs over many projects.

4) It encourages the development of tools which can reduce the redundant efforts common to software development activities.

5) It supports both the technical and the management aspects of system development in a single facility.

6) It establishes a basis for a host-independent software development system.

This section describes how the authors' study of the software development process and recent work in the field of software engineering has led to the rationale for the SEF.

Software Development

Software development presents the challenge of making progress in a constantly changing environment. We are familiar