SSA/AIDS: A GRAPHIC, INTERACTIVE SYSTEM FOR STRUCTURED SYSTEMS ANALYSIS

Richard D. Hoffman, Linda N. Harris and Brett W. Bickham
Exxon Corporation
Florham Park, New Jersey

ABSTRACT

Computer application requirements collection and analysis occurs at Exxon through a methodology called Structured Systems Analysis (SSA), which emphasizes the modeling of systems via a series of diagrams. In this paper, we discuss a tool, SSA/AIDS, which provides computer-based support for SSA via an electronic drawing board. We begin by describing the applications development life-cycle in general, and briefly describe Exxon's history with methodologies and tools which support this life-cycle. We concentrate on a specific task in the cycle, requirements collection, and present SSA as a methodology for accomplishing this task. We show how SSA is supported by SSA/AIDS, and conclude with a discussion of our plans to extend SSA/AIDS so that it supports the entire applications development life-cycle.

KEYWORDS: requirements collection, software engineering environments, interactive graphic systems, computer-assisted development.

Introduction

In the past decade, the growth of software and graphics systems devoted to the automation of industrial design processes has been enormous. The Computer Assisted Design (CAD) station has proven to be a great productivity aid, and has become so ubiquitous that we are usually surprised when we encounter a manufacturer without one. Yet there is no commercially available package which provides similar aid to developers of applications software. Computer professionals have spent many hundreds of work-years automating the work of others, while only lately turning our attentions to the problem of helping themselves.

The amount of interest shown in computer-assisted software development systems has recently begun to increase, as industry has begun to appreciate the productivity gains which can be attained through the use of such systems. This paper describes some of the work we have been doing at Exxon to increase the productivity of our own applications developers and, in particular discusses a graphic, interactive design tool, SSA/AIDS, which automates part of the work of systems modeling and requirements analysis. For a good survey of related work being done by others, see Hünke [2].

The applications development life-cycle

The applications development life-cycle in use at Exxon consists of six phases through which all software application development projects pass. These phases include:

- Scoping (high-level problem definition)
- Exploration (detailed problem definition)
- Specification
- Design
- Development
- Startup

Starting in the early seventies, Exxon began to develop a comprehensive set of methodologies to support these phases and to improve the productivity of applications developers. These methodologies emphasized the use of diagrams and tables wherever possible to enhance communications between project analysts and their clients.

The first of these methodologies was Program Structure Technology (PST), in which detailed design is accomplished through a series of hierarchical diagrams such as the one in figure 1. Based on a technique developed by Jackson [3], PST improved productivity by standardizing the program design process, shortening the time required to code and test applications, and guaranteeing thorough documentation.

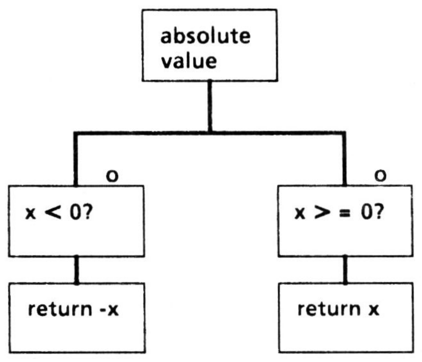

FIGURE 1:
The Program Structure Diagram, a hierarchical (tree) diagram employed by PST.

When, in the late seventies, we began to develop tools to support the methodologies, we started with PST. The result (based upon a prototype developed at an Exxon affiliate) was PST/AIDS, a tool to assist the user with the PST methodology. PST/AIDS permitted the analyst to draw his diagrams on a Tektronix 4014 storage tube, and offered some syntax checking of the diagrams, and a primitive but useful code generation facility. It provided a fixed pattern of potential boxes which could be made visible and connected to produce trees (hierarchical diagrams). Its command set included commands to delete, copy and move subtrees, which significantly simplified diagram creation and editing. Most commands were implemented by cross-hair movement and one or two keystrokes.

For more information about PST and PST/AIDS, see Menard [4]. Both PST and PST/AIDS support only the Design and Development phases of the life -cycle, as do most commercial productivity aids. To improve productivity earlier in the life-cycle, we developed a methodology to model systems, and thus assist in the scoping and exploration phases of the life-cycle. The remainder of this paper deals with that methodology, and the tool which supports it.

Structured Systems Analysis

When analyzing a system or business in the scoping and exploration phases of an application, it is useful to start by developing a "system model". This is a description of a system which is

- thorough,
- clear,
- concise,
- unambiguous,

and from which one may abstract requirements. The problem of constructing models which satisfy these criteria continues to plague analysts and applications developers. Most methodologies which currently exist for modeling systems either fail to satisfy one or more of the four criteria or lack the generality to model a broad class of systems.

At Exxon, we have developed a systems modeling methodology, Structured Systems Analysis (SSA), based on the work of Jackson [3], Yourdon [1], and others. The methodology employs a set of four annotated diagrams and some supplementary information (consiting of tabular and textual material) to describe a system in terms of:

- the decomposition of the system into functions which create, transform or destroy data;
- the flow of information and material between these functions;
- the decomposition of functions into activities; and
- the decomposition of data aggregates into elements.

The diagrams are created and modified as a result of a series of interviews between the analyst and the client, and through the study of supporting data. Three of the diagrams have the same simple tree structure of the PST diagrams (see figure 2). The information flow diagram (listed second above) has a more complicated structure, consisting of a network of symbols which may be interconnected in many ways (see figure 3).

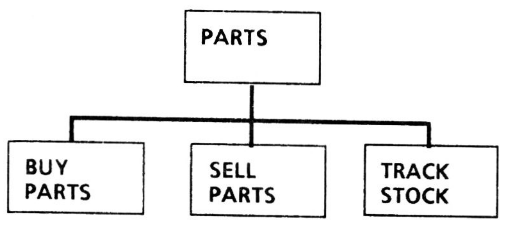

FIGURE 2:
An example of the Global Model, one of three tree diagrams employed by SSA.

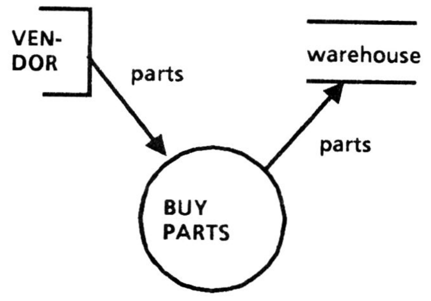

FIGURE 3:
Part of an Information Flow Diagram.

When completed, the SSA diagrams provide a system model which satisfies the necessary criteria: the model is thorough, clear, concise and unambiguous. The methodology is flexible enough to model a very large class of systems, and can do so at varying levels of detail, as required by the client and the task at hand. For more information on SSA and its sources, see Mendes [5].

The need for automation

SSA has achieved a fairly wide usage within Exxon-- most of our affiliates have adopted it as their standard for systems analysis. We have had considerable success using SSA to model and analyze systems, but we have also realized that the technique could be made even more productive.

The SSA diagrams posess simple structures, but can sometimes be quite complicated and thus prove difficult to draw. A diagram modeling a section of a large, complex system may require many pages. Furthermore, since the SSA process is usually iterative, the analyst may find it necessary to make several revisions of each diagram before the model is complete.

The diagrams, which are the main vehicle for communications between the analyst and his client, must be clear and comprehensible, so revisions usually entail a complete redrawing of the diagram. By the time a model is complete, each diagram may have been redrawn by the analyst many times, and probably at least once (for presentation purposes) by a graphic artist.

And as with any system model, one has the problem of storing and maintaining large amounts of data in some easily retrievable form. All these things force analysts to concern themselves with matters other than the content of their models, and are potential barriers to productivity.

One finds analogous problems in the tasks of creating and editing large amounts of code or text. This suggests a solution in the form of a computer-based diagram editor.

Design directions

In SSA/AIDS, we intended to provide a tool which would assist the analyst in all phases of the SSA process. We also wished to encourage the use of SSA by making it easier and more profitable to apply than alternative analysis techniques. In this sense, we intended SSA/AIDS to be a "lure" for SSA. And finally, we wanted to ensure that SSA/AIDS would be compatible with future tools.

We arrived at three major requirements which drove the design of SSA/AIDS. They were:

- Completeness -- SSA/AIDS must facilitate the on-line computer representation of all SSA diagrams and supporting material. Furthermore, the representation must posess high graphic quality, and must be automatically reproducible on paper. Anything less than this forces the user to make a choice between doing the job partially on the computer or entirely by hand, and this is a choice that will sometimes go against the computer, no matter how sophisticated the tool.

- Convenience -- SSA/AIDS must be user-friendly, self-explanatory and error resistant, as well as easier and faster than the manual process. These requirements should guide the design of all interactive tools, but take on special importance in a tool which will mostly be employed by infrequent or occasional users of the computer. We designed SSA/AIDS so that a user of SSA could log on to one of several host operating systems, invoke SSA/AIDS, and proceed immediately without recourse to an instruction manual, and without fear of destroying his own work or the work of others.

- Compatibility -- SSA/AIDS will serve as the foundation for future tools. We wished to design its editors and underlying data structures so that it could easily be extended to handle new types of diagrams and information. This forced us to keep the tool as general as possible, within the context of the applications development life-cycle. A secondary concern here was the minimization of confusion and frustration to analysts who use both PST/AIDS and SSA/AIDS.

We will now show how these requirements were met by the current implementation of the tool.

High-level Operation

SSA/AIDS runs in several different IBM mainframe environments (including TSO), and uses the Tektronix 4014 for graphic input and display. It operates in four modes, highly differentiated from each other to minimize confusion. These modes consist of:

- General (commands to delete, copy, rename and analyze diagrams);

- Tree Edit (commands to construct and edit the diagrams pictured in figure 2);

- Network Edit (commands to construct and edit the diagrams pictured in figure 3); and

- Help (a series of tutorial panels on each command and matters in general).

The tree editor has the same screen appearance and command structure as the diagram editor in PST/AIDS. Although the network editor required a fairly different set of commands, they were kept similar wherever possible, once again to minimize confusion for previous PST/AIDS users.

In both edit modes, the screen is divided into two areas: a drawing board, where the diagrams are constructed and edited, and a system response window, where warning and error messages are displayed, and where the user is sometimes prompted for information which will not appear in the diagrams. These two areas are separated by a banner that identifies the diagram being edited, and its type.

Figures 4 and 5 depict these two screens. Notice that the banners are very different, so that the user can identify his environment immediately. The general and help modes are characterized by interactive dialogue, and are differentiated through their prompts.

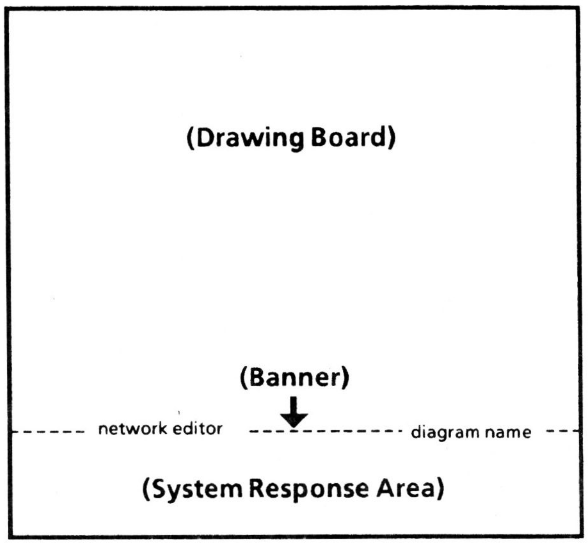

FIGURE 4:
Network editor screen.

In each of the four modes, every user action leads to some kind of visible system response, so that the user always knows what mode he is in, and whether the system is responding. Table 1 offers a summary of the four modes.

A discussion of the detailed workings of SSA/AIDS is not within the scope of a short paper. However, in the next few paragraphs we highlight some of the more interesting of these details with a few examples from the design of the network editor.

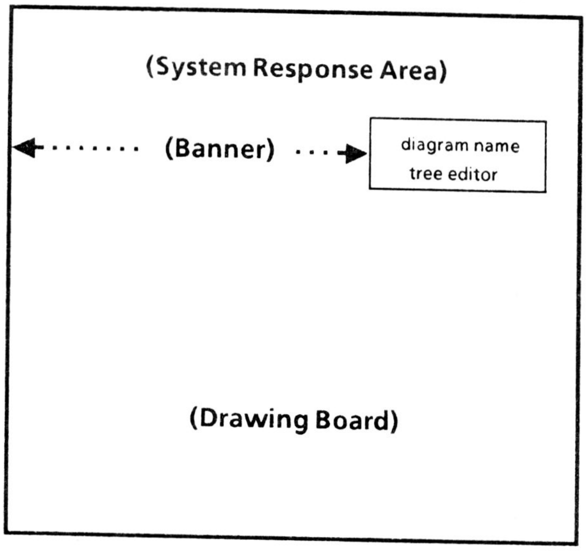

FIGURE 5:
Tree editor screen.

The network editor

The most important (and interesting, from a computer-graphics point of view) sub-task in the design of SSA/AIDS was the design of the network editor. Although it will have other purposes in the future, the network editor currently exists only to facilitate the information flow diagram of SSA. Since this diagram is often the key to a successful SSA model, we took extra care to ensure that our three requirements were satisfied in the network editor.

To undersand our approach, one must first know a little about the structure of the information flow diagram (IFD). Looking at figure 3 again, one sees that there are several types of nodes in an IFD and that lines of any orientation may join these nodes. Nodes may also be joined by arcs. Arrowheads are used to indicate the direction of flow. Annotations are placed inside the nodes and alongside the connecting joins. Join annotations may be placed anywhere, as long as the association between the join and the annotation is visually clear.

SSA prescribes rules by which IFDs are to be constructed. For example, certain nodes must be uniquely named, and there must be no unconnected nodes. On the other hand, SSA offers only guidelines-- no firm rules-- to limit the amount

			Entry to other modes			
Mode	Purpose	Mode cues	General	Tree Edit	Net. Edit	Help
General	Management of diagrams (create, copy, delete, etc.)	Prompt for command, no crosshairs		E command, tree qualifier	E command, network qualifier	H command
Tree Edit	Construction and editing of tree diagrams	crosshairs, banner at top	Q (quit) or X (exit) command		No direct transfer	H command
Network Edit	Construction and editing of network diagrams	crosshairs, banner at bottom	Q (quit) or X (exit) command	No direct transfer		H command
Help	Obtaining information about the system and its commands	Prompt for help, no crosshairs	null line returns user to mode from which help mode was entered			

TABLE 1:
Summary of SSA/AIDS command modes

of data on a single IFD. Consequently, they can get very crowded. Nevertheless, diagram appearance is very important, since a diagram which is too cluttered will probably not be easily understood.

Here are a few of the design issues which faced us in the design of the network editor:

- How can the editor determine the optimum location for a join annotation?
- How can the editor ensure a "good-looking" diagram?
- How can the editor ensure a "correct" diagram?

The answer to each of these questions was "It can't." There was no scheme for the automatic placement of text for which we could not immediately conceive of several counter-examples in which the automatic placement was decidedly wrong. Similarly, there was no way to automatically reformat the diagrams in a way that would always please the user.

On the other hand, it would have been possible to force the analyst to conform to the rules of SSA, but not desirable. In the early stages of an SSA project, most analysts want to experiment with their diagrams, and often violate SSA rules while creating their model. These analysts eventually correct their diagrams, but would find a tool which deprived them of the freedom to occasionally violate a rule very frustrating.

In answer to these and other questions, we adopted the principle of "user control". We resolved to give the user as much control over his own work as was possible, and to make it easy for him to do anything that he might reasonably wish to do within the context of SSA. If he then desired to maintain diagrams which were incorrect or unclear, that was his responsibility. In this way, SSA/AIDS facilitates the different ways in which different analysts have come to use SSA. The next few paragraphs provide some examples of this principle as applied to the three questions raised above.

Annotation placement

A user requests a join by typing "J" over the two nodes which he wishes to connect. SSA/AIDS then displays a prompt in the system response area, requesting that the user designate the starting location of the annotating text, and the text itself. To ensure that the user sees the prompt, a warning beep sounds as the prompt is issued. To guard against the possibility of the user still not seeing the prompt, SSA/AIDS tests the next location which the user designates for reasonable proximity to the join. If the location is too far from the join, SSA/AIDS assumes that the user has forgotten that he needs to specify the location, and reminds him in the system response area. If the user actually wants to leave the join unannotated, he enters a null line after specifying the location.

Should it later transpire that the user has placed the annotation in a poor position (perhaps because it is too near the annotations of succeeding joins), he may use a command which moves the text on the diagram without requiring it to be re-entered. Figures 6a and 6b show an example of this.

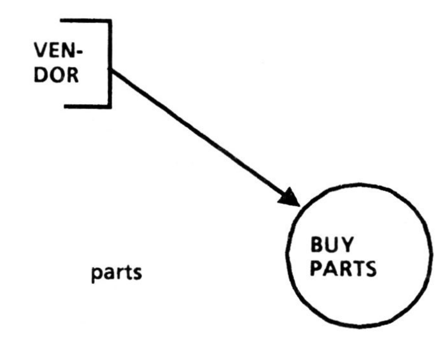

FIGURE 6a:
A typical join. The text "parts" is in the wrong place.

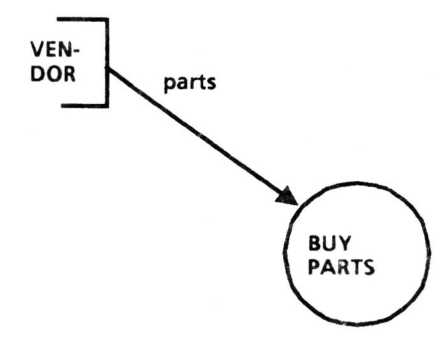

FIGURE 6b:
The join from figure 6a, three keystrokes later.

Diagram appearance

SSA/AIDS does not permit nodes to overlap. There are no other rules to affect the appearance of an IFD. Nodes may lie as close to each other as the user desires, and joins may cross over nodes and other joins. A join annotation may be placed almost anywhere, even if it obscures other text. However. the "Move" command allows the user to eliminate any of these overlaps that are not actually necessary. In practice, most users' first approach to a diagram is a rough draft in which they discover new entities, which they record anywhere in the diagram. Then, as the relations between entities becomes clearer, the move command, which preserves joins, proves a very convenient way to clean up the diagram.

Figures 7a and 7b show an example of the move command. Annotating text is automatically positioned (via an algorithm which keeps the relative locations of text and join constant), but can be easily re-positioned by the user. The move command has proved to be the most powerful command in the SSA/AIDS repertoire since, in a few keystrokes, it accomplishes a task which previously required the redrawing of an entire diagram. No user of the tool so far has felt the need for any other kind of reformatting device.

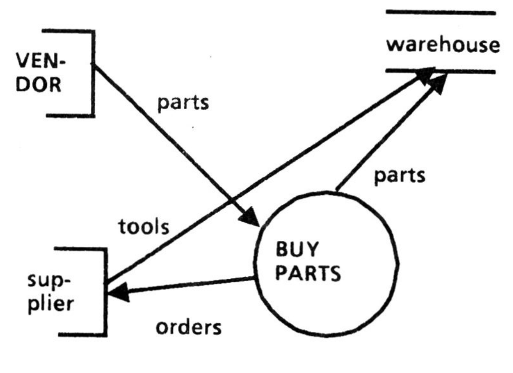

FIGURE 7a:
A cluttered portion of an Information Flow Diagram.

Rule enforcement

SSA/AIDS detects a number of SSA rules violations. When they occur, the user receives a warning message in the system response area, but is never forced to comply. Thus the user is free to break the rules if he sees fit. The warning, however, prevents inadvertent violations, and also helps instruct new users of SSA.

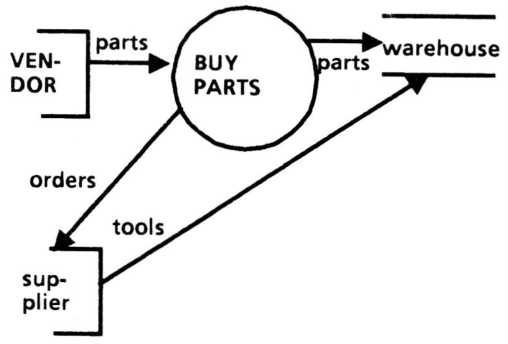

FIGURE 7b:
The diagram from figure 7a, two keystrokes later.

Future enhancements

SSA/AIDS has already begun to help analysts with the SSA process. Users have indicated that SSA/AIDS has not only significantly speeded up their work, but that it has also helped them to produce work of better quality, since it frees them from worrying about the appearance of their diagrams, and allows them to concentrate on the content instead.

We are now using SSA/AIDS to analyze the SSA process and the current version of SSA/AIDS itself in preparation for a major enhancement of the tool. This enhancement will include:

- Extensions to the existing functionality. Most SSA/AIDS features currently only deal with one diagram at a time; we will add features to check diagrams against each other for consistency and correctness, and to allow more information exchange between diagrams. We will also add features to analyze the diagrams for potential trouble-spots in the system being modeled.

- New functionality. SSA/AIDS does not yet support the tabular and textual elements of SSA. There are also elements of other methodologies which will soon be included, so that the tool will cover other phases of the applications development life-cycle besides scoping and exploration.

- Better environment. We are experimenting with prototypes of SSA/AIDS on terminals other than the 4014, such as IBM's 3279 color graphic terminal. We are also looking into the possibility of running SSA/AIDS on a dedicated micro-computer.

Our ultimate goal is to provide a set of tools which support the entire applications development life-cycle, and to put these tools within easy reach of all our analysts. SSA/AIDS, which is already improving the productivity of our analysts, even when applied to itself, is a major step toward that goal.

BIBLIOGRAPHY

[1]. Constantine, L. and E. Yourdon, *Structured Design*, (Englewood Cliffs, NJ: Prentice-Hall, 1979).

[2]. Hünke, H., *Software Engineering Environments*, (Amsterdam: North-Holland, 1980).

[3]. Jackson, M., *Principles of Program Design*, (London: Academic Press, 1975).

[4]. Menard, J., "Exxon's Experience with the Michael Jackson Design Method," *Database*, Winter-Spring 1980, pp. 88-92 (reprinted from *Proceedings of the Application Development Symposium*, Monterey, California, October 1979, pp. 143-8).

[5]. Mendes, K., "Structured Systems Analysis: A Technique to Define Business Requirements," *Sloan Management Review*, Summer 1980, volume 21, number 4, pp. 51-63.