The positive benefits resulting from use of PSL/PSA compensated for the negative experiences. Plans have been made to utilize the system to describe software requirements for other military systems.

ACKNOWLEDGMENT

The work reported in this paper was accomplished by the Martin Marietta Corporation under contract to the Air Force System Command’s Space and Missile Systems Organization (SAMSO). I would like to thank them for their assistance in the preparation of this report.

REFERENCES


EXPERIENCE WITH SADT—Donn Combelic

BACKGROUND

SADT, Structured Analysis and Design Technique, is a registered trademark of SofTech Inc., Waltham, Mass. ITT has used the “Structured Analysis” part of SofTech’s SADT since early 1974. In mid-1975 we began to develop for real-time switching software our own structured design technique, called FP2 for Functions-Processes-Flowgrams-Programs, based upon precepts and syntax of Structured Analysis. Thus the technique we are now using for analysis and design is called SA/FP2.

GENERALITIES

The principal basic ideas of the technique are: determine the “what” before the “how”, decomposition from the top down to reveal successive levels of detail, output in the form of diagrams each of which gives a limited amount of detail, each diagram is critiqued in writing by one or more persons other than the author of the diagram, needed information unknown to an author is obtained by interviews with outside experts. Each diagram is comprised of boxes that represent “activities” interconnected by arrows that represent “data” used by an activity for input, output and control. A box plus its arrows constitutes the “context” of that activity—it is that (bounded) context which is decomposed to understand and show more detail in a diagram at the next level.

OVERVIEW OF SA/FP2.

SA/FP2 is carried out in five phases, one of which is concurrent with two of the others. The first phase is that of Structured Analysis (SA); the remainder are those of design, that is, FP2. A brief summary of each of the five phases is given in the following paragraphs.

Structured analysis phase

SA is ideally applied to the total system. However, in most applications we have applied it only to real-time switching software. In such a case, the primary inputs are a list of customer requirements plus functional specifications of the telephone hardware of the system. The output is a Functional Requirements Model (FRM) in the form of a set of activity diagrams many of which are accompanied by a page or two of explanatory text and definitions of terms. The FRM shows what functions the software must contribute in addition to those of the telephone hardware in order to fulfill the customer functional requirements. To the greatest extent possible, software design considerations, such as data layouts, scheduling, priorities, handling of queues, buffers and computer peripherals, are kept out of the FRM. Thus the FRM emphasizes the “what,” not the “how”.

Transfer phase

This is the first phase of design. Its principal inputs are the FRM and the software design constraints. Typical constraints are: choice and arrangement of computers, computer peripherals, programming language, requirements for traffic handling, engineerability, extensions, maintainability, etc. The outputs of the transfer phase are a high level data layout model and a single level “action group” model. (A software action is defined as a sequence of instructions which, once started, can run to completion without waiting because all needed inputs were available at its start.) For each action group, a convenient set of one or more contiguous activities, along with the data arrows at the boundaries, is selected at an appropriate level of detail from the FRM and transferred (as a single box) to the action group model. The action groups are interconnected as the corresponding sets of activities were interconnected in the FRM. The high level data layout model is developed before and during the transfer procedure. The transfer phase is complete when all activities of the FRM have been accounted for in the action group model.
Action group decomposition phase

All action groups are decomposed to the level of individual actions. The output is a set of activity diagrams where each box at the lowest level of detail represents an individual action. An additional output is further detail of the data layout. During this and the preceding phase the decomposition rules are the same as for the SA phase, but the SA syntax is augmented to handle action starts and completions.

Flowgram phase

Each action is decomposed, according to its implicit control flow sequence, down to the level of individual routines, each of which appears as a separate box on the lowest level diagrams. The control flow is shown on each diagram in a special syntax, hence the name “flowgram.” The output of this phase is a “flowgram model” for each action. The previous syntax is augmented to handle control flow. It turns out that when the control flow sequence and the individual routines are coded the resulting set is a structured program. Thus there is a structured program for each action.

Coordinator phase

The coordinator is that software which, among other things, starts all actions and to which all actions return upon completion. It thus includes the functions of scheduling, handling of queues and management of memory. It is convenient to include within the coordinator the treatment of interrupts and the handling of telephone and computer peripherals. It is interesting to note that none of these functions relate directly to the customer’s basic functional requirements, rather they all depend on the nature of the system. The functional requirements for the coordinator begin to emerge as early as the SA phase, become more clear by the end of the transfer phase, but cannot be known completely until the action group decomposition phase is finished. By that time the coordinator can be completely specified and designed. Note that the techniques described for the preceding phases can be applied to the analysis and design of the coordinator.

EARLY EXPERIENCE—1974-1976

Development of the FP2 design methodology reached the point where it could be used in practice only at the beginning of 1977. Thus all our prior experience was limited to Structured Analysis as taught by SofTech and refined by ITT and SofTech together. We adopted SA in early 1974 for two main reasons. First, it provided a disciplined way of understanding requirements in detail before starting design. Second, it offered a method which promoted teamwork. The latter was a particularly difficult problem on some of our projects in Europe where a team would consists of members with widely varying experience from up to eight different ITT companies speaking six different languages. Of the approximately twenty ITT projects where SA has been used, all but one are in Europe.

Strong and weak points

A partial list, derived from our early experience with SA, is as follows:

- SA estimated to decrease overall software development cost by at least 20 percent and significantly improve software quality—estimated 2 to 10 times less bugs found during integration testing, varies with project.
- Very definitely promoted teamwork.
- Hard to think all the time in purely functional terms.
- The written comments (by other than the author) required for each diagram resulted in continual review, in effect “walkthroughs.” (Note: Commentators should normally be other authors in the same team.)
- Interviews of outside experts proved efficient method of obtaining specialized information.
- Forced making high level decisions early, thus providing a sound basis for later lower level decisions.
- Encouraged agreement on requirements before start of design.
- Lack of follow-through on design methodology (later overcome by FP2) was bothersome.
- Permitted non-software people to understand the contributions of software functions to system operation.
- Much more useful information per sheet (diagram) than with documents in prose.
- Provided easy way of measuring progress during analysis.
- SA excellent for many applications other than real-time switching software, but space does not permit elaboration.

Some mistakes made and lessons learned

A partial list follows:

- Method was oversold in the beginning as a panacea.
- Proper use of SA requires a fundamental change in mental outlook—difficulties of achieving such a change were underestimated.
- Mere “training” is inadequate—“education” is required.
- Potential authors must be selected on the basis of intelligence and willingness to try a new way rather than purely on experience.
- Constructive critics of the method are to be cherished, but destructive critics must be eliminated from the SA group as soon as they surface.
- Method takes much more time before design starts than previous methods. Inevitable if requirements to be thor-