USING OBJECT ORIENTED STRUCTURED DEVELOPMENT TO IMPLEMENT A HYBRID SYSTEM

Federico Vazquez
Computer Sciences Corporation
15245 Shady Grove Road
Rockville, Md 20850

Abstract

Object Oriented Analysis, Design and Programming are increasingly impacting the development approach to Computer Systems. Although these methodologies are not new, their use in industry is increasing and they have had a large impact in both System Analysis and System Design. Some people think that object oriented techniques will be the new predominant methodologies used in the 90s and there is no relationship with Structured Design. Others believe that there is indeed a relationship between Structured Analysis and Object Oriented Design. During my work with the development of computer systems I have found a certain degree of compatibility between Object Oriented Analysis and Structured Analysis. I did not however find compatibility between Structured Design and Object Oriented Design. This paper deals with the use of both techniques in a successful system development effort with a hybrid approach. It is possible to work with hybrid systems where Object Oriented and Structured development are combined and complement each other.

Keywords: Object Oriented Development, Object Oriented Design, Object Oriented Analysis, Object Oriented Programming, Structured Development, Structured Analysis, Structured Design, Object Oriented Structured Development.

Introduction

Object Oriented Programming emerged as a term associated with the development of Smalltalk in 1982. Object Oriented Design was attributed to Grady Booch, although it was first introduced by Russell Abbott [Abb83]. OOD addresses preliminary design, simulation, and detailed design. Object Oriented Analysis is best typified by the work of Coad and Yourdon and is a method of analysis that examines requirements from the view of classes and objects found in the Problem Domain. Structured Analysis was introduced by Tom De Marco and improved by Yourdon. Structured Design is a Hierarchical Decomposition Method introduced by Yourdon and Constantine. Object Oriented Structured Development techniques have been introduced by several authors [Kha89], [War89], [Was89] and although a general methodology is not yet complete, the general principles for a top-down bottom-up approach are common and broadly accepted as a development process.

Object Oriented Structure Development

Structured Analysis (SA), Structured Design (SD) and Structured Programming (SP) are collectively known as Structured Development (SDV). All the many versions of SDV are based on a philosophy of system development that analyzes the system from a functional point of view. Constantine describes the point of view as the main features of software that are of interest to the analyst, designer or programmer. The main features are: what functions or tasks the software must perform, what subfunctions or subtasks are needed to complete the overall functions, what pieces or component parts will perform various functions, how those functions will be performed. The structured project life cycle is describe in Figure 1.

Object Oriented Development (OODV) composed of Object Oriented Analysis (OOA), Object Oriented Design (OOD) and Object Oriented Programming (OOP) as a development philosophy has its origin in OOP and evolved bottom up, from programming to design to requirements analysis. Thus, it is helpful to be familiar with OOP to understand the OODV paradigm.

Constantine describes the Object Oriented paradigm. In this paradigm,

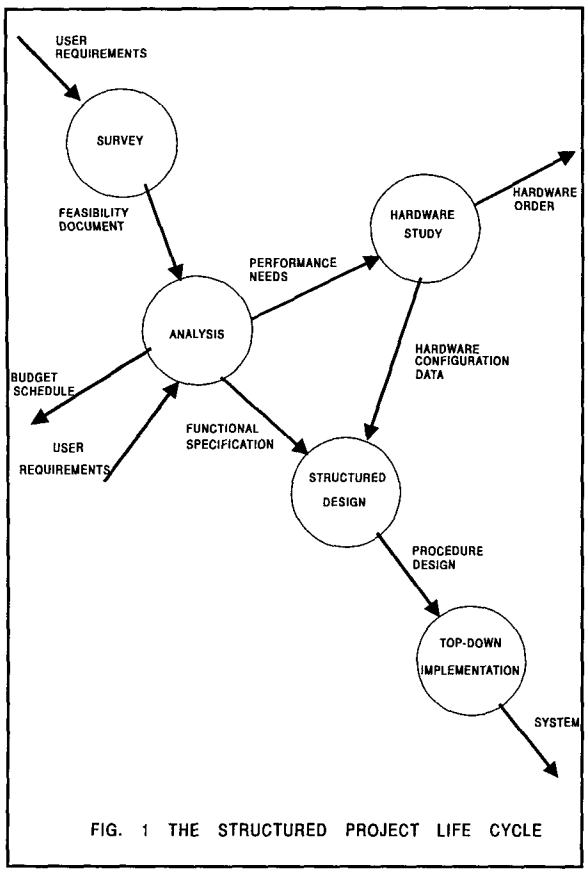

FIG. 1 THE STRUCTURED PROJECT LIFE CYCLE

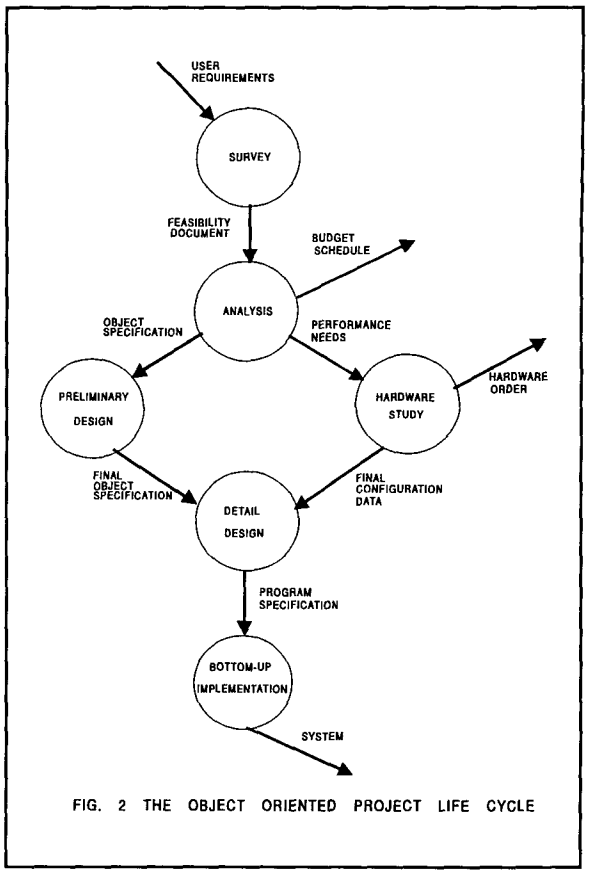

FIG. 2 THE OBJECT ORIENTED PROJECT LIFE CYCLE

data are analized first and procedures complement this analysis; functions are associated with related data. Problems and applications are looked upon as consisting of interrelated classes of real objects characterized by their common attributes, the rules they obey and the functions or operations defined on them. The project life cycle for OODV is show in figure 2.

The SD techniques are generally associated with a top-down development approach, whereas OODV is essentially a bottom-up approach. The "top" of a system structure contains control modules representing the activation of procedures of the overall capabilities. At the "bottom" are the basic facilities for defining and manipulating the data, and for hiding its structure from the rest of the system. These concepts are embodied in the idea of the "object". OODV thus facilities a natural bottom-up organization of software.

Object Oriented Structured Design (OOSD) synthesizes top-down and bottom-up approaches to software design and various mixed approaches. The top-down design uses functional decomposition to partition a system into modules. Structured Design supports functional decomposition that uses structure charts as a design representation.

OOD identifies classes (templates to create objects) that are appropriate for a given system. These classes are often derived from classes that have been used in previous designs, and thereby support reuse. Classes serve as building blocks in the overall design structure. The foundation of OOSD is the Entity Relationship Model. This approach permits designers to add to their experience with SD and evolve toward OOSD.

Entity-Relationship Model

The Software Engineering problems are grouped on the basis of similar characteristics. These are called problem domains. Fortunately, there

are not many domains; a simple classification can be real time systems, on-line systems, reactive systems and concurrent systems.

The entity relationship approach uses the Entity Relationship Model [CHE76] to categorize information from the real world problem domain. It recognizes that the system needs to be considered at the logical level. This information is conveyed by defining the entities in the domain, the interrelationships of those entities, and attributes passed by the entities. These concepts must ultimately be mapped into a design that can be implemented in computer systems.

After reviewing several OOD methods including Booch Object Oriented Design [BOO86], General Object Oriented Software Development (GOOD) [SEI87], Object Oriented Analysis Systems Modeling [SHL88], Object Oriented Analysis Design [COA90], it was found that the Entity-Relationship Model was the foundation of these Object Oriented Methods. Context Diagram is the foundation of Structure Analysis. Although the Entity-Relationship Diagram is not the same as the Context Diagram, it can be considered as a Context Diagram. With this assumption we have the Entity-Relationship Diagram as the initial and common diagram for OOA and SA.

After we have an abstraction of the problem domain with the Entity-Relationship Diagram we are able to continue the OOA and SA, because we base the analysis in the same Entity Relationship Diagram, we expect to find similarities between OOA and SA.

Object Oriented and Structure Analysis

There are many articles and books that describe how to identify both objects and the nature of the objects. Shaler and Mellor have made a contribution in this area, providing tools and concepts for enumerating various categories of potential classes.

Objects are the heart of OOA; the behavior of the object, the way that the objects are related, the attributes of the objects, and their derived services constitute the OOA. SA uses Data Flow Diagrams (DFD). When you create a DFD you are considering in the functionality of the Design, the inputs to a process and the corresponding output of this process.

It is true that process (bubbles) in the DFD are different from the objects in OA, but something that has been shown is the relationship between parts of the DFD and set of objects in the OOA. When you represent an object, you have to analyze the behavior of the objects, identify the operations and identify relationships. The State Transition Diagram and DFD are helpful diagrams for this purpose.

SA is used to identify and clarify objects in OOA. SA and OOA are complementary of each other. The user is able to relate a State Transition Diagram and DFD to OOA Diagram. They are also compatible when they are derived from the same set of specifications as often is the case.

Object Oriented and Structure Design

The next step in the project life cycle is the Design. The mapping from OOA to OOD is not isomorphic. The preliminary OOD is an extension of OOA (more elaborate OOA diagram). In the Detail OOD, the method addresses static and dynamic behavior, parent child and seniority hierarchies, object class decomposition and is tied or related to the target language. If the target language is Ada or Object Pascal, Booch uses one set of notations; if the target language is Smalltalk, CLOS, C++ Booch uses another set of notations. Although Yourdon and Codd try to create a Design independent of the language, they acknowledge that Detail Design should be related with the target language. Ada is not an Object Oriented Language. Booch calls it an Object Based Language, because the use of inheritance and Dynamic Binding is not possible. There are several articles like the one in [Don90]; in this article Donaldson gives a way to implement these properties of object oriented languages, but the code needed to implement these properties will increase complexity and size of the

programs. The best way to resolve these problems is to acknowledge the Ada deficiencies with relation to the Object Oriented languages and just exploit the facts of encapsulation and data abstraction. Ada is an ideal language to implement these properties. The OOA and OOD are an effective approach for Ada implementation of a system and the Booch notation is very suitable.

SD can be derived from SA and the implementation in a procedure language is straight forward. The notation used for the representation of the SD is widely accepted and the Central Transformation is a fact in the process of conversion between SA and SD [YOU89].

There is no relationship between SD and OOD because the philosophies are different. C and C++ are different, they share some common properties inherent to programming languages, but the way to code the same system is different from C to C++ and consequently is a different between Procedure and Object Oriented languages.

Object Oriented Structure Implementation

Traffic Count is a hybrid system (implemented using different programming languages). It counts the number of arrivals, departures and type of airplanes for an air traffic control facility or airport during a certain period of time. The Traffic Count System contains requirement specifications (RS) written by Software Engineering. The RS explain the functionality of the system. An example of requirements specifications is shown in Figure 3. These RS are the first step in the development of the system and they allow to have a general overview of the design of the system.

After Analyzing the RS the System Analysts develop Entity- Relationship Models describing data to be recorded (Flight Data Recorded). Then additional requirements are derived. Data Base files need to be created to keep a historical track of the traffic counts. A User Interface (panels) is required to allow the generation of Traffic Count Reports and to allow user inputs of selection criteria. A series of events is identified which are necessary to produce the Traffic Count Reports. These events include the matching of information of certain types of airplanes (air carrier, general aviation, military, air taxi). Finally requirements for the production of Traffic Count Reports are defined as: types of airplanes, report time periods, facility identifiers and airports. A very simplified Entity-Relationship Model is shown in Figure 4.

TRAFFIC COUNT FUNCTION

The Traffic Count function accepts traffic count information and maintains the traffic count data by aircraft operation category and flight type sub-category for each sector within a facility. Counts for, airport operations, and the use of airways are also maintained.

The Traffic Count function provides the automatic traffic count display, statistical summary reports and the generation of traffic count forms.
The following table provides the decomposition of this Function.

Function type	Function Name
Display	Traffic Count Display
Reports	Statistical Summary Reports
Generation	Generation of Traffic Count Forms

Fig. 3 Requirements Specifications

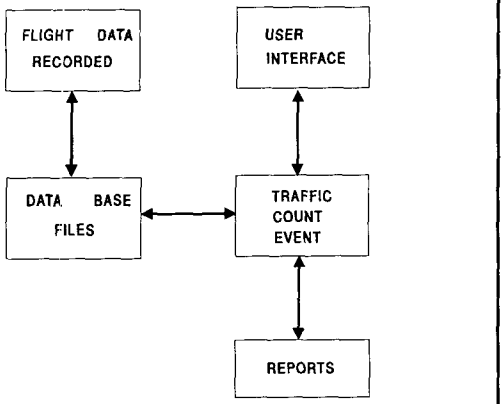

FIG. 4 Entity Relationship Model for Traffic Count Events

After the Entity-Relation Model is completed, the second part of the Analysis deals with the Detailed Analysis of the Traffic Count System. It is usually an OOSD implementation. Some of the diagrams from the Entity-Relationship Model are expanded and the search for objects, attributes of the objects, object operations and object interfaces begins. Figure 5 describes a simplified OOA for traffic count.

The functional approach of the implementation is also considered because the behavioral view from SA describes the expected behavior of the system. Ada is not a true Object Oriented language, the programmers are not true Object Oriented programmers and the implementation of the system is developed using several target languages. In the DFD shown in Figure 6, there is a simplified functional representation of the Traffic Count System. As mentioned before, every part of the DFD has an association in the OOA, and it helps to understand behavioral capabilities of the objects.

In the Object Oriented Structured development, the design usually consists of several phases. First a General Design is developed with iterative draft and latter Group Walk Throughs. When this process is completed a detailed design inspection is held. Figure 7

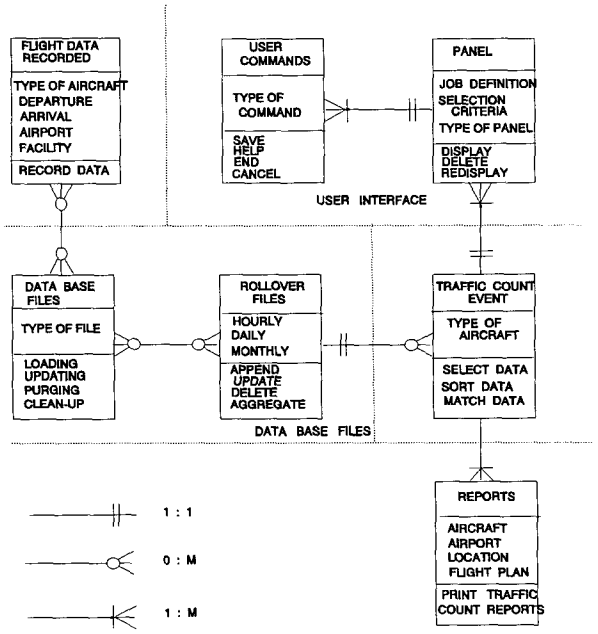

FIG 5. OBJECT ORIENTED ANALYSIS FOR TRAFFIC COUNT

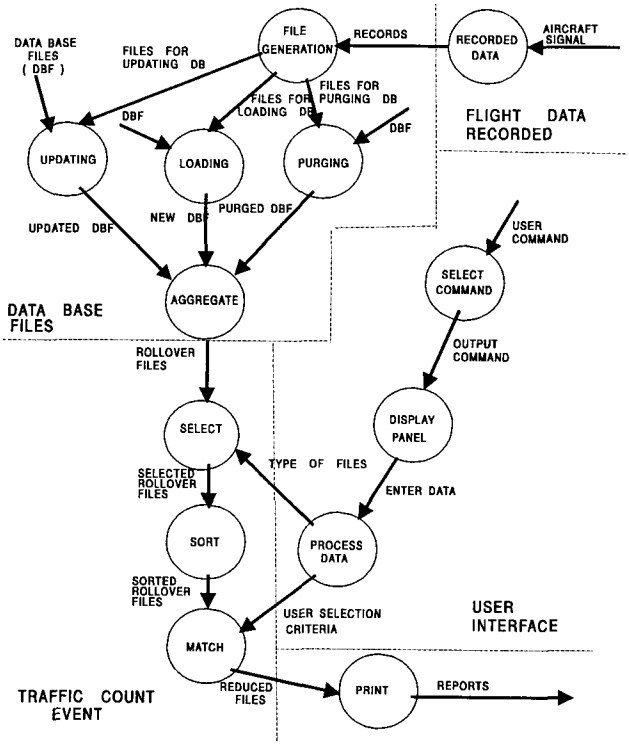

FIG. 6 STRUCTURE ANALYSIS FOR TRAFFIC COUNT

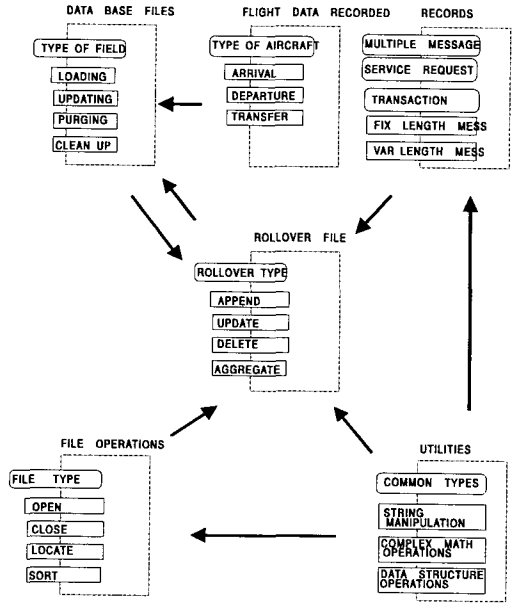

FIG. 7 OBJECT ORIENTED DESIGN FOR DATA BASE AND FLIGHT RECORDED DATA

shows a simplified detailed design for "Data Base" and "Flight Recorded Data" of the Traffic Count Reporting Area. We use the Booch notation to reference the OOD. The relationships among objects (interfaces), the attributes, and the object operations are clearly shown in Figure 7. This part is implemented in Ada and the conversion from a Detailed Design to a Program Description Language is isomorphic with help from the DFD.

The SD for the user interface is shown in Figure 8. This part was targeted for implementation using REXX. As REXX is not an Object Oriented language, it does not make sense to use an Object Oriented approach here. In Figure 9 we have a simplified structure chart for the Traffic Count Event. This figure was created with a tool to convert DFD to structure charts. After a central transformation is chosen and it was implemented using Statistical Package for Social Sciences (SPSS).

The Detail Design includes a Program Description Language (PDL) for each object in the OOD and a PDL for each module in the Structure Chart are generated. A rigorous inspection is held to review the PDL, and after the PDL is approved, the coding of the module begins. After code is completed another inspection is held and the programs are unit tested and later string tested.

In Figure 10, we have an example of a Traffic Count Report program. As is obvious from the figure, this program (called a macro) is just one procedure containing the inputs to the Object (procedure) and the outputs from the object (procedure). The report generated from this procedure is shown in Figure 11.

Object Oriented and Structure Development Compatibility

James Martin in his book "Object Oriented Analysis and Design" [MAR92] introduces several new methods where he mixes Object Oriented and Structured Approach (like Object Data flow Diagrams). Ward has written a brief tutorial showing that there is no inherent conflict between the two

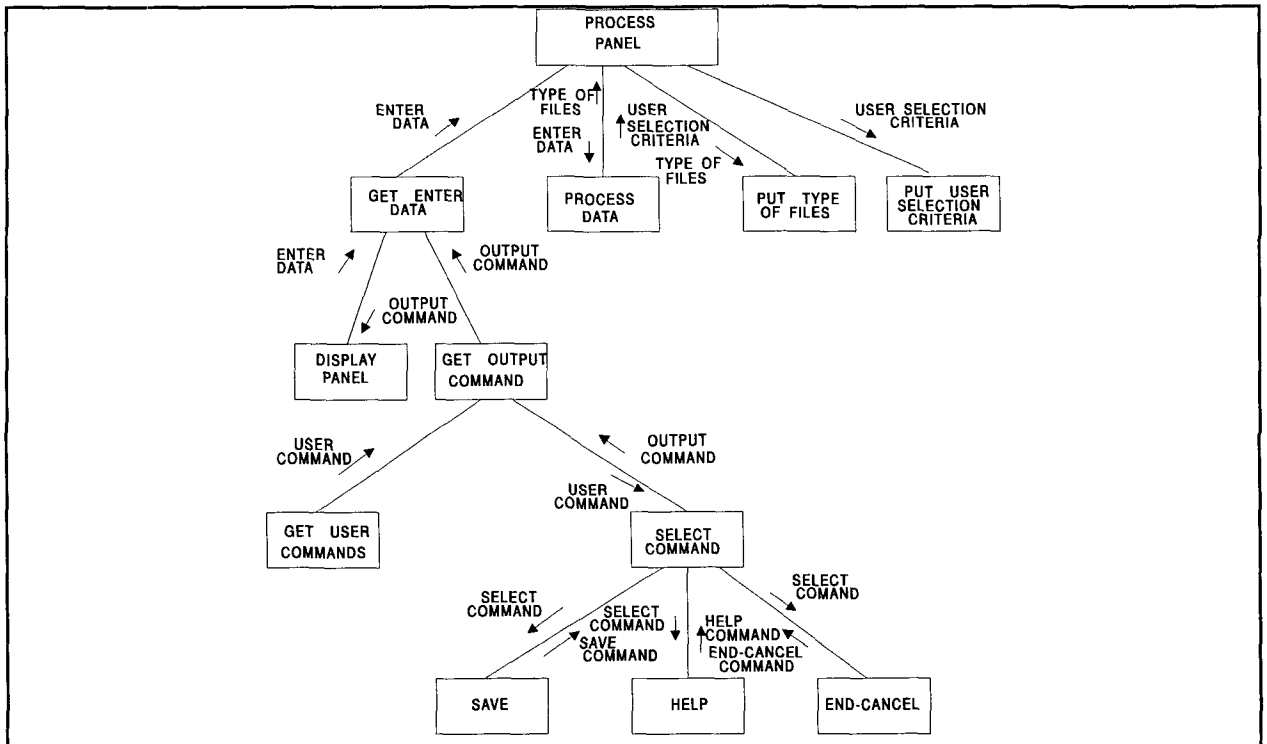

FIG. 8 STRUCTURE DESIGN FOR USER INTERFACE

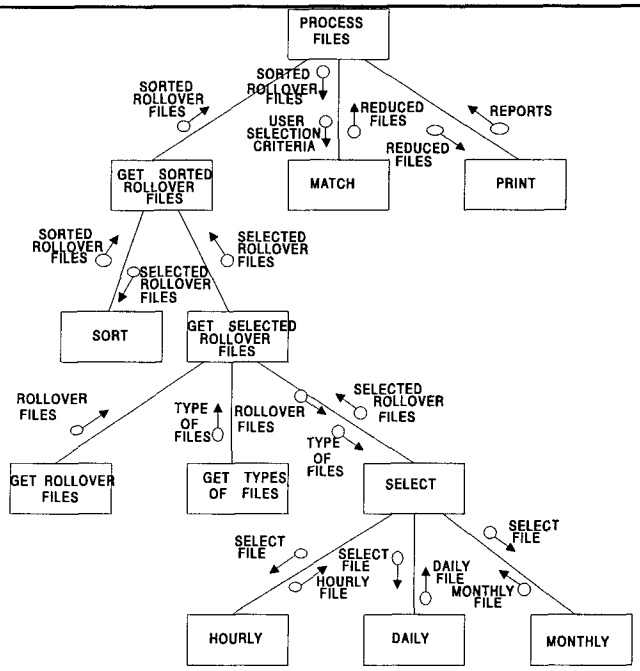

FIG. 9 STRUCTURE DESIGN FOR TRAFFIC COUNT EVENT

```
* DECLARE FORMATS TO BE USED IN THE REPORT
- FORMATS BEGTIME ENDTIME (DATETIME20) / SORTSTR (A14)  CURTIME (TIME8)
* PRINT THE REPORT USING THE MERGED FILE
- REPORT FORMAT = AUTOMATIC LIST MISSING ' '
   /STRING = DEPARTUR ( ' ' DEPARTURES ' ' )
             ARRIVALS  ( ' ' ARRIVALS ' ' )
             OVER      ( ' ' OVER ' ' )

   /VARS = ACDEP 'AC'
           GADEP 'GA'
           ATDEP 'AT'
           MIDEP 'MI'
           TODEP 'TOTAL'
             ACARR 'AC'
           GAARR 'GA"
           ATARR 'AT'
           MIARR 'MI'
           TOARR 'TOTAL'
             ACOVE 'AC'
           GAOVE 'GA"
           ATOVE 'AT'
           MIOVE 'MI'
           TOOVE 'TOTAL'
   /TITLE= LEFT ' '  ')DATE ' ')CURTIME '
           ' SELECTION CRITERIA '
           ' START DATE, TIME          :  )BEGTIME'
           ' STOP DATE, TIME           :  ) ENDTIME'
           ' FLIGHT TYPE               :  ) FLIGHTY'
           ' TRAFFIC COUNT AREA        :  )TRACOA'
           ' TRAFFIC COUNT AREA ID     :  )TRACOI'
           ' RECORDING TYPE            :  )RECOT'
   /CENTER 'HOURLY TRAFFIC COUNT REPORT'
    RIGHT  ' '  ' PAGE '  'DRATCC-TCH'
```

FIG. 10 SPSS PROGRAM FOR TRAFFIC COUNT REPORTS

HOURLY TRAFFIC COUNT REPORT

01/01/92
11 : 01

PAGE 01
DRATCC-TCH

SELECTION CRITERIA
START DATE, TIME : 06/24/89, 210000
STOP DATE, TIME : 06/24/89, 230000
FLIGHT TYPE : DOMESTIC
TRAFFIC COUNT AREA : FACILITY
TRAFFIC COUNT AREA ID : ZAB
RECORDING TYPE : SYSTEM GENERATED

	DEPARTURES					ARRIVALS				OVERS			
TIME	AC	GA	AT	MI	TOTAC	GA	AT	MI	TOTAC	GA	AT	MI	TOT
21:00	10	08	12	05	45 20	15	05	05	45 10	08	10	07	35
22:00	25	18	15	08	66 30	20	15	10	75 11	14	12	10	47
23:00	55	45	28	16	144 72	35	25	15	14745	30	22	12	109

AC = AIR CARRIER
GA = GENERAL AVIATION
AT = AIR TAXI
MI = MILITARY
TOT = TOTAL

FIG. 11 OUTPUT OF THE SPSS PROGRAM

approaches [WAR89]. Jalote proposes an "extended object oriented design methodology" which incorporates a top-down, step-wise refinement approach [JAL89]. Bailin describes a method for combining Structured Analysis with the Object Oriented approach for requirements specifications [BAI89]. Constantine has written two papers that address the topic of the integration of the methods.

The application of a similar methodology as used in this article is reported by [KHA89], [WAR89] and [SHU91]. It has been found that other authors do not accept compatibility, like Firesmith [FIR91] who enumerates several unfounded risks (risk associated with complex data flow, requirements traceability and changing the paradigms in the middle of the project). Brodman considers OOSD as an SA and SD technique with artificial Object flavor [Bro91]. Reed who only sees complementary design between Jackson System Design and OOD and others who could consider DFD as a tool to analyze the behavior, identify operations and interconnections among objects.

Outlook

In the development of this system there are many professionals with different backgrounds. Many have worked with different consultant companies and used less common methodologies. All these approaches are brought to the table in some way as the system is being developed. It is interesting to note that several programmers do not have an intrinsic knowledge of the Object Oriented Paradigm and consequently they do not produce Object Oriented Programs. They do however have a knowledge of the programming language. They do effectively use Ada to achieve information hiding. Using non-object oriented techniques they develop procedures for SPSS and REXX. They have the ability to visualize and integrate various parts of the system that have a non-object and object oriented approach in their design and programming. This proof that Object

Oriented and Structured Development are complementary of each other. This paper describes a process to implement an information system with SD as a complementary technique for OOD and it shows that a real problem needs to be addressed with several design techniques.

Unfortunately the mathematical theory behind OOD and SD is almost null. Several authors are using Petri Nets because Petri Nets have a mathematical base; other authors including myself are trying to formalize these methodologies and find a common mathematical theory that pushes the theory of knowledge for System Development to a higher level and facilitates the evolution of more coherent and useable methods.

Acknowledgment. The author wishes to thanks Myron Shear for their valuable comments in this paper.

References

[Abb83] Abbott, R. J., "Program Design by Informal English Descriptions", Communications of the ACM, Vol. 26 No. 11, (1983) pp. 882-895.

[Bai89] Bailin, S. C., "An object oriented requirements specification method", CACM, Vol. 32, No. 4 (1989) pp. 608-623.

[Boo86] Booch, G., "Object-Oriented Development", IEEE transactions on Software Engineering, Vol. 12 No. 2 (1986) pp. 211-221.

[Boy87] Boyd, S., "Object-Oriented Design and PAMELA", Ada letters, Vol. VII, No. 4 (1987) pp. 68-78.

[Bro91] Brookman, D., "SA/SD vs OOD", Ada letters, Vol. XI, No. 9 (1991) pp. 96-99.

[Che76] Chen, P. "The Entity- Relationship Model -- Toward a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, No. 1 (1976), pp. 9-36.

[Che91] Chen, M.J. and C.B. Chung, "Restructuring Operations for Data-Flow Diagrams", Soft. Eng. J. (1991), pp. 181-195.

[Coa90] Coad, P., E. Yourdon, Object-Oriented Analysis, Englewood Cliffs, NJ, Yourdon Press, 1990.

[Don90] Donaldson, C.M., "Dynamic Binding and Inheritance in an Object- Oriented Ada Design", J. of Pascal, Ada & Modula-2 (1990) pp. 13-18.

[Fir91] Firesmith, D. "Structured Analysis and Object-Oriented Development are not compatible", Ada letters, Vol. XI, No. 9 (1991) pp. 56-66.

[Gan82] Gane, T. and C. Sarson, Structured System Analysis, Mc Donnell Douglas, 1982.

[Jal89] Jalote, P. "Functional Refinement and Nested Objects for Object-Oriented Design", IEEE Transactions on Soft. Eng (March 1989).

[Kha89] Khalsa, G.K., "Using Object Modeling to transform Structured Analysis into Object Oriented Design", Proceedings of the sixth Washington Ada Symposium (1989) pp. 201-212.

[Kha88] Khalsa, G.K., Using three System Perspectives to transform Structured Analysis into Object-Oriented Design. Arizona State University, Temple Arizona.

[Mar84] Martin, J., System Design From Probably Correct Constructs: The Beginnings of true Software Engineering, Englewood, Cliffs, NJ, Prentice Hall 1984.

[Mar92] Martin, J. and James J. Odell, Object-Oriented Analysis & Design, Englewood, Cliffs, NJ, Prentice Hall, 1992.

[Loy90] Loy, P.H. "A comparison of Object-Oriented and Structured Development Methods", Soft. Eng. Notes (1990) pp. 44-48.

[Pre87] Pressman, R.S., Software Engineering A Practitioner's Approach, Singapore, Mc Graw-Hill, 1987.

[Sod91] Sodhi, J., Software Engineering Methods, Management and Case Tools, Blue Ridge Summit, Pa, TAB Professional and Reference Books 1991.

[Sei87] Seidewitz, E. and M. Tracks, "Towards a General Object-Oriented Software Development Methodology", Ada Letters Vol. VII No. 4 (1987) pp. 54-67.

[Shl88] Shlaer, S. and S.J. Mellor, Object-Oriented Systems Analysis: Modeling the World in Data, Englewood Cliffs, NJ, Prentice Hall 1988.

[Shl92] Shlaer, S. and S.J. Mellor, Object Life Cycles Modeling the World in States, Englewood Cliffs NJ, Yourdon Press, 1990.

[Shu91] Shumate, K. "Structured Analysis and Object-Oriented Design are Compatible", Ada Letters, Vol. XI, No. 4 (1991) pp. 78-90.

[Sut91] Sutcliffe, A.G., "Object-Oriented Systems Development: Survey of Structured Methods", Information and Software Technology, Vol. 33, No. 6 (1991) pp. 433-442.

[SPS90] SPSS Inc, SPSS Reference Guide, SPSS Inc., 1990.

[War89] Ward, P.T. "How to Integrate Object Orientation with Structured Analysis and Design", IEEE Software, (1989) pp. 74-82.

[Was89] Wasserman, A.I., P.A. Pircher and R.J. Muller, "An Object-Oriented Structured Design Method for Code Generation" SIGSOFT Software Engineering Notes, Vol. 14, No. 1 (1989), pp. 32-55.

[You89] Yourdon, E., Modern Structured Analysis, Englewood Cliffs, NJ, Prentice Hall, 1989.