

INTRODUCTION TO ESSENTIAL SYSTEMS ANALYSIS

John W. Satzinger
Department of Management
Terry College of Business
The University of Georgia
Athens, GA 30602-6256

Introduction

The *structured systems analysis* methodology has dominated formal college level training in systems analysis for over a decade. Structured analysis has evolved slightly over the years (mainly to integrate advances in data modeling techniques), but the use of *data flow diagrams*, the *data dictionary* and *process descriptions* remains largely unchanged in most textbooks. In addition, the process followed often includes constructing multiple system models (including the existing physical system model) prior to constructing a logical model to represent new system requirements.

Alternative systems analysis tools and techniques are emerging to replace structured analysis, including both business area analysis from *information engineering* (Martin, 1990) and *object-oriented analysis* in support of object-oriented design and programming (Coad & Yourdon, 1991). These approaches clearly deserve attention in information systems courses. At the very least, time spent on structured analysis must be reduced to allow for coverage of these methods. However, some argue that structured analysis offers little and should be completely replaced with newer more rigorous methods when training information system analysts.

This paper discusses the use of a refinement to structured analysis, called *essential systems analysis*, which adds rigor to the use of structured analysis modeling tools and streamlines the analysis process. It is argued here that the use of essential systems analysis complements the newer methodologies and has a valuable place in formal systems training and in practice. First, the evolution of structured analysis is reviewed. Then, the concepts of essential systems analysis are summarized. Next, a simple example using essential systems analysis is described. Finally, the benefits of the essential systems analysis approach and strategies for its use are discussed.

The Evolution of Structured Analysis

The structured systems analysis method evolved from the consulting work of Ed Yourdon and associates in the late 1970s (DeMarco, 1978; Gane & Sarson, 1977). Structured analysis, like structured design, is often referred to as the Yourdon approach. Prior to structured systems analysis, a variety of approaches to systems analysis were used. Systems analysis of the flow of information through an organization predates computers and goes back to industrial engineering and process flow charts developed early in this century. Documents and document flows became increasingly important after 1920, and document flow charts came into use. By the 1950s,

Copyright © 1993 by John W. Satzinger, Ph.D.. All rights reserved.

A preliminary version of this paper was published in the *Proceedings of the Seventh Annual Conference of the International Academy for Information Management*, Dallas Texas, December 11-13, 1992.

computers were made part of work systems, and charts and diagrams were developed to document computer requirements and designs (Cougar, 1973).

As of the early 1970s, Cougar (1973) referenced 59 system development techniques or tools that were proposed or used for systems analysis. The systems development lifecycle (SDLC) used today has not changed much since then, and systems analysis meant understanding the existing system and defining the logical requirements for a new system prior to designing the system, as it does today. Cougar discussed tools and techniques for automating system development, and even proposed that it might be possible to define logical requirements for a system and have a computer tool generate the programs automatically (as of 1973).

Despite the many techniques and tools that existed before structured analysis, many information systems analysis and design textbooks imply that systems analysts produced the user requirements for a system in a cumbersome narrative form which no one was willing to read or use. The result was the system designers did not base the design of the system on the documented requirements, and the documented requirements probably were incorrect or incomplete since the users probably did not actually read them before approving them. This somewhat biased description appears in DeMarco's (1978) book on structured analysis, where he argues for the use of his tools and techniques. Colter (1984) refers to structured analysis as part of the *fourth generation of systems analysis* techniques, and acknowledges that it is quite different from previous approaches, even though previous approaches did rely on graphical models.

Structured analysis uses *functional decomposition, data flow diagrams, process descriptions,* and a *data dictionary* to model information systems. Three models were created during the analysis process. First, a *physical model of the existing system* was created using data flow diagrams and supporting documentation. Then, the physical details were removed from the diagrams and supporting documentation to produce a *logical model of the existing system*. Finally, changes were made and functions were added to the logical model of the existing system to produce a *logical model of the new system*. This model became a complete specification of the requirements for the new system.

As database management and data modeling techniques appeared, structured analysis evolved to integrate *data modeling* into the analysis process. Data stores on the data flow diagrams came to represent tables (in relational data models) or entities (in entity-relationship data models). A side effect of integrating data modeling was the more radical change in the requirements model when data stores were not simply "improved" in the new system but actually reorganized. Models of the existing system became much less useful when creating a model of the new system requirements.

Essential Systems Analysis Concepts

Essential systems analysis was first described in 1984 and used in Yourdon analysis and design seminars beginning in 1985 (McMenamin & Palmer, 1984; Ward & Mellor, 1985). By this time, industry reactions to structured analysis were mixed. Many complained that the process of modeling the existing physical system, stripping away physical artifacts to create a logical model

of the existing system, and then modifying the logical model of the existing system to incorporate improvements and new requirements was too time consuming in practice. Also, many unnecessary artifacts from the existing system were often retained since the existing physical system was the foundation for the new system. In addition, although data flow diagrams were useful for communicating processing requirements, there were few guidelines for using them to model the system -- either for decomposing the system or for deciding when the diagrams were adequately detailed. Separate analysts working on the same system were likely to develop models that differed considerably. Further, integration with structured system design using transform and transaction analysis was weak because very detailed data flow diagrams were required (c.f., Page-Jones, 1980).

The essential systems analysis methodology addresses many of these problems, although it has not yet been incorporated into most textbooks that feature systems analysis methods (for an exception see Yourdon, 1989). With essential systems analysis the existing system is studied but not modeled. Instead, attention is more quickly focused on the logical requirements for the new system. Only one system model is produced -- *the essential model*. Additionally, guidelines are provided for decomposing the system based on the concept of *events*. This heuristic improves the chance that separate analysts will produce similar models of the same system. Finally, the concept of *perfect technology* is used to keep the focus on logical rather than physical requirements.

An information system is viewed as a *planned response system*. Planned activities in response to events in the environment are designed in to a system. Requirements for a system are defined by considering the events that occur to which the system must respond regardless of the way it is eventually implemented. There are two types of events -- external and temporal. *External events* occur in the environment beyond the system's control, such as when a customer wants to place an order. *Temporal events* occur based on the arrival of a point in time, such as when it is time to produce monthly bills or when a payment is 10 days late.

When an event occurs the system must have a way of knowing about the event's occurrence. This is called the *stimulus*. For external events, the stimulus is an input data flow. For temporal events, the stimulus is the arrival of the point in time (sometimes shown on data flow diagrams as a dashed line control flow). When the event occurs, the system is required to carry out some activity, called an *essential activity*. In doing so, the system may create outputs to the environment, which are called *responses*.

A system may be required to respond to events that might not be initially anticipated. These are called *ad hoc responses*. It is useful to try to separately list as many of these events as possible because they can give the analyst a feel for the possible maintenance and enhancement needs the system may face over time. These also include ad hoc requests for information, which the system should be able to provide on request.

The concept of *perfect technology* helps the analyst focus on the underlying logical requirements. Many aspects of a physical system are designed in to compensate for limitations of technology and people. These often end up in the physical design as information system controls or as design compromises based on limitations in capacity. When modeling an existing physical system, much

of the effort goes into modeling these processes. The underlying logical user requirements sometimes become obscured by these physical details. To help keep the analyst focused on the underlying requirements, it is assumed that the system will be implemented with perfect technology -- human and computer processors never make mistakes, data storage devices never lose data, and speed and capacity are never constrained. Therefore, when events are considered, only those the system must respond to when implemented assuming perfect technology are included in the essential model. A true logical model is therefore produced.

Essential Systems Analysis Process

The first step in the process is to produce the *environmental model*. This is a model showing the system interacting with the environment. It consists of a list of events and a *context data flow diagram*. An *event table* is useful for organizing all of the information needed to derive the context diagram. It lists the event, the type of event, the stimulus, the source of the stimulus, the essential activity required, the response(s) and the destination of each response. The stimulus can be a data flow or the arrival of a point in time. A response, if there is one, is a data flow.

The event list serves as the basis for partitioning the system into loosely coupled modules. An event triggering an essential activity should be included if 1) a planned activity is required when the event occurs, 2) the activity is required even if it is assumed the system will be implemented using perfect technology, and 3) the system is at rest after carrying out the activity when triggered by the event. The context diagram is drawn to graphically depict the system's interactions with the environment.

The second step is to produce the *behavioral model*. It models the system's behavior internally and is therefore a refinement of the environmental model. It consists of a *data model* (called essential memory), a set of *data flow diagrams*, a *data dictionary* and *process descriptions*. The data model is derived by considering what information must be remembered by the system to carry out essential activities. An entity-relationship model is recommended. As entities are identified, additional events might be discovered which the system must know about to keep data current. A distinction is therefore made between *fundamental activities*, which contribute to the underlying purpose of the system, and *custodial activities*, which are required to keep the essential memory up to date. Additional events can be added to the event table and the context diagram can be updated iteratively.

Although the data dictionary, process descriptions and data model are used conventionally, the way data flow diagrams are applied when developing the behavioral model differs from structured analysis. First, one data flow diagram is produced for each event. Each data flow diagram (called a *DFD fragment*) contains one process which represents all of the system's activity in response to the one event. Data stores, data flows and external entities (sources and destinations) can be shown for the one activity. Once all DFD fragments are completed, they are combined to form *diagram 0*. In this respect, a bottom up rather than top down process is followed. And events are used to decompose (or partition) the system. Diagram 0 is therefore called the *event partitioned system model*.

The event partitioning heuristic assures that a similarly decomposed set of diagrams will be produced by separate development teams. Additionally, the processes are very loosely coupled since each is a self contained model of one system activity. The event partitioned system model, therefore, has the characteristic of no process to process data flows -- all communication from one process to another occurs through data stores. These are great improvements in rigor for data flow diagram modeling. The event partitioned system model, supported by the data model, data dictionary and process descriptions, graphically communicates the essential requirements for the system, showing all required system components -- inputs, processes, storage and outputs.

Each process on diagram 0 can be further decomposed if desired, as is traditional in structured analysis, or it can be simply described using *structured English*. Those activities that are highly interactive might be better modeled using *state transition diagrams* or working *prototypes*. Those that produce reports or other outputs may be better modeled using a *fourth generation language syntax*. The resulting integrated processing and data storage model, which is quite compact, is still useful as a communication tool even when used as a supplemental model along with information engineering models. Structured analysis based CASE tools can be readily used with essential systems analysis.

Structured system design has also been refined based on essential systems analysis concepts (Page-Jones, 1988). A transaction in *transaction analysis* has been redefined to be the system's required computer processing when triggered by an event. One *structure chart* is produced for the automated part of each essential activity, so the event partitioning heuristic carries over into system design, and detailed data flow diagrams are not required to achieve some integration with system design and implementation.

In summary, essential systems analysis produces a logical model of both processing and data storage requirements using the same graphical tools and supporting documentation as structured analysis. However, the event partitioning heuristic and the concept of perfect technology streamlines the analysis process and increases the rigor of model. In addition, integration with structured design is improved.

System Example

To illustrate the essential systems analysis approach, a simplified example of a *public library video tape check out system* is described below. This example assumes the system is to be a separate, but integrated module of the library system. Simplifying assumptions are:

- o The book check out system already exists and it creates and maintains records of library patrons
- o A bookkeeping system already exists and it collects payments for library fines
- o Ordering new video tapes is handled by the purchasing department

The essential requirements for the tape check out system are described in terms of events. First, a patron will want to check out tapes and then later return them. Sometimes a patron will neglect to return a tape by the due date and needs a reminder. Occasionally the tape will not be returned

at all and should be considered lost. New tapes will be received by the library from time to time. Library management will want information about tape check out activity about once a month.

These six events are listed in the event table shown in Figure 1. Three events are external and three are temporal. A patron checking out a tape, a patron returning a tape, and the library receiving a new tape are external events. They can occur at any time. When these events occur, the system must respond. When a patron returns a tape, the tape might be on time, overdue or previously assumed lost. Since the system knows the status of each checked out tape, it can produce the appropriate response (return of a damaged tape is ignored to keep the example simple).

Figure 1
Simplified Event Table For Library Video Tape Check Out System

Event	Type	Stimulus	Source	Essential Activity	Response	Destination
1. Patron wants to check out tape	Ext	Check out request	Patron	Check out tape	Check out agreement	Patron
2. Patron returns tape	Ext	Tape return	Patron	Accept returned tape	Patron late charge notice	Patron
					Bookkeeping late charge notice	Bookkeeping
3. Tape is overdue	Temp	"5 days past due date"		Produce late tape notice	Late tape notice	Patron
4. Overdue tape is assumed lost	Temp	"20 days past due date"		Produce lost tape notice	Patron lost tape charge notice	Patron
					Bookkeeping lost tape charge notice	Bookkeeping
5. New tape is received	Ext	New tape details	Purchasing	Add new tape		
6. Time to produce monthly tape activity report	Temp	"Last day of each month"		Produce activity report	Monthly activity report	Management

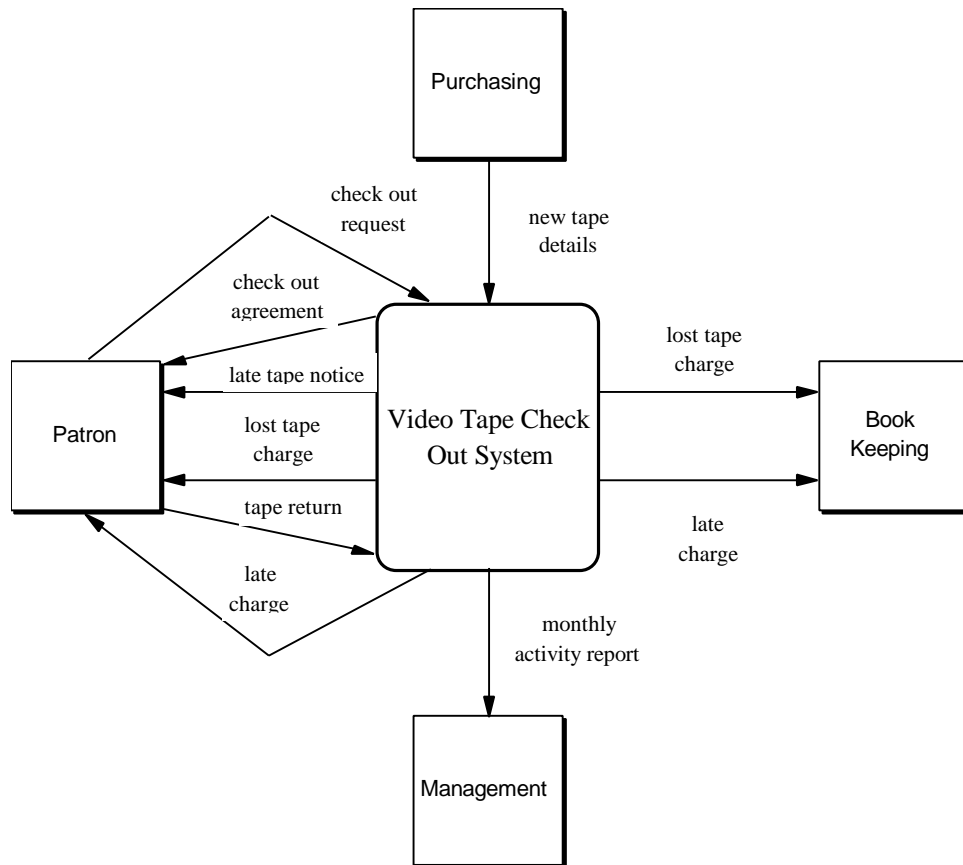
A tape is overdue at a predefined time relative to the due date, and a tape is assumed lost after an additional predefined time relative to the due date. These are temporal events that trigger a late notice for an overdue tape and a lost tape charge for a lost tape. Since these times are set relative to the due date, and tapes are due any day the library is open, the system should be able to respond daily. Some might argue that late notices and lost tape charges could be generated as a

batch once a week to be more convenient for the system. However, the perfect technology assumption would argue there is no reason for batching. A system with perfect technology could produce a response the second a particular tape becomes overdue.

The third temporal event triggers the production of an activity report for library management. Here a system with perfect technology is capable of continuously updating tape check out activity in a summary form. However, management does not want to deal with this information continuously. Therefore, the event is defined to occur at a fixed point in time -- once a month. This follows from the assumption that the system is perfect, but external entities are not. Therefore, the system must respond in a way that is convenient for the external entity. The report is not produce this way to be convenient for the system.

The rest of the event table defines the stimulus, the source of the stimulus, the essential activity that is triggered by the event, the response (if any), and the destination for the response(s). The stimulus is either a data flow (check out request, tape return, or new tape details) or a point in time (5 days overdue, 20 days overdue, or the last day of the month). There is a source for data flows but not for points in time. The essential activities triggered by the event are named following data flow diagram process naming conventions.

Figure 2: Context Diagram



The context diagram can be produced directly from the event table. Four external entities, three input data flows and seven output data flows are included in the model. This is shown in Figure 2. The context diagram might be produced concurrently with the event table as each event is uncovered or even as the first step in the process depending on the preference of the analyst (Yourdon, 1989).

The event table also provides the foundation for the behavioral model. The data elements captured from the input data flows and produced for the output data flows provide clues to the attributes of objects required in the data model. The data model is shown in Figure 3. Once the data model is produced, each essential activity is modeled as a DFD fragment which integrates processing requirements and data storage requirements in one self-contained module. The data flow definitions and process descriptions are then completed for the one activity. The six resulting DFD fragments are shown in Figure 4. This approach allows the analyst to focus on one essential activity at a time. Once all essential activities are understood and modeled, they are combined to form the event partitioned system model (diagram 0) shown in Figure 5.

Benefits of Essential Systems Analysis

Essential systems analysis is quite usable in practice despite the introduction of many new (and perhaps exotic) terms and concepts. Once past this terminology, the resulting data flow diagrams, data model, data dictionary, and process descriptions still provide the familiar graphic and non-redundant set of documentation. But fewer diagrams are drawn, attention is focused on one essential activity at a time, and unnecessary artifacts from the existing physical system are minimized. The model can be completed in less time and the tools and techniques can be taught in less time than structured analysis.

There are many benefits to the use of essential systems analysis to complement both information engineering and object-oriented analysis. It is argued here that the essential model should be produced early in the planning stages of the system project. This is particularly feasible if the data model used in the essential model is kept at the subject database level to further streamline the process. Then, depending on the characteristics of the system or the development environment chosen, information engineering, object-oriented analysis and design, structured design or prototyping methodologies can be used to complete the system.

Figure 3: Data Model (Essential Memory)

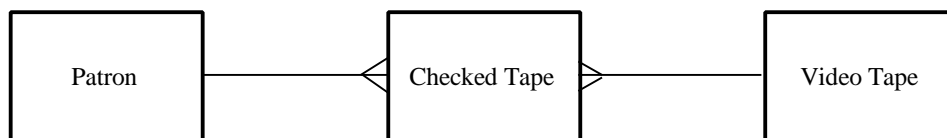
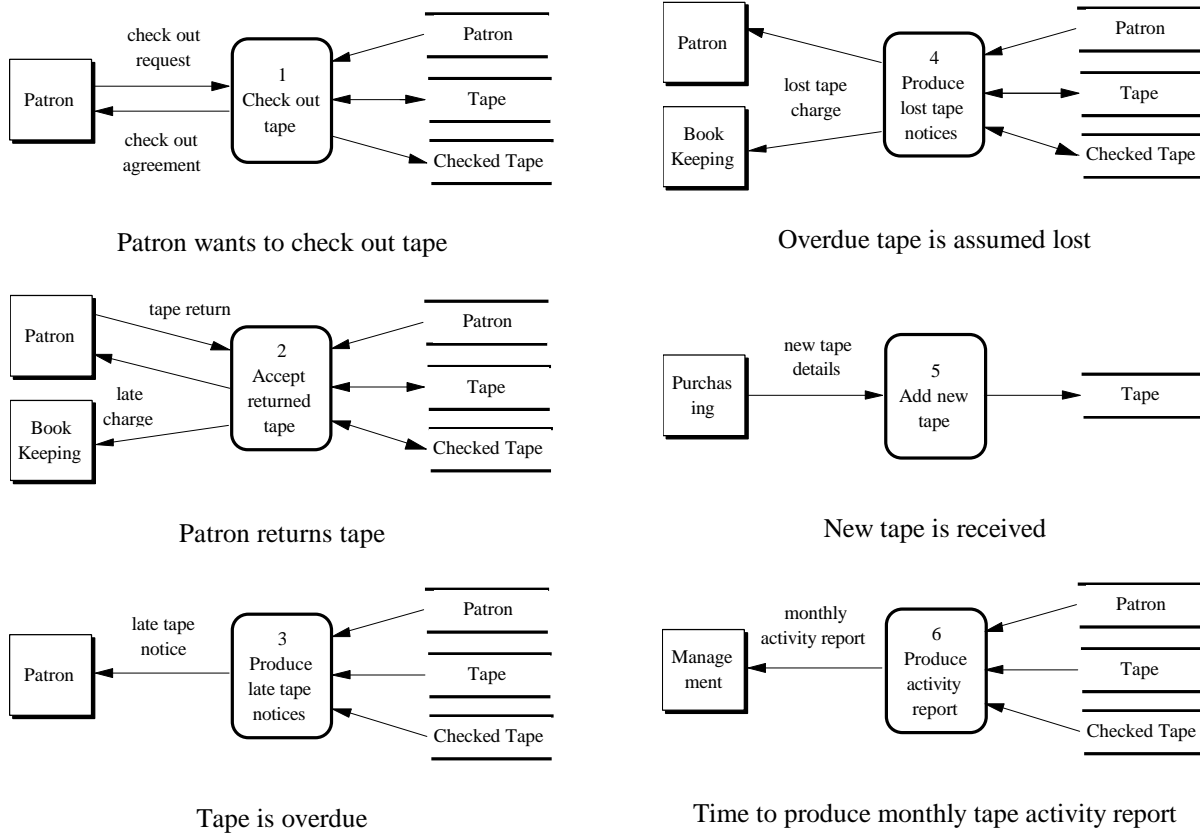


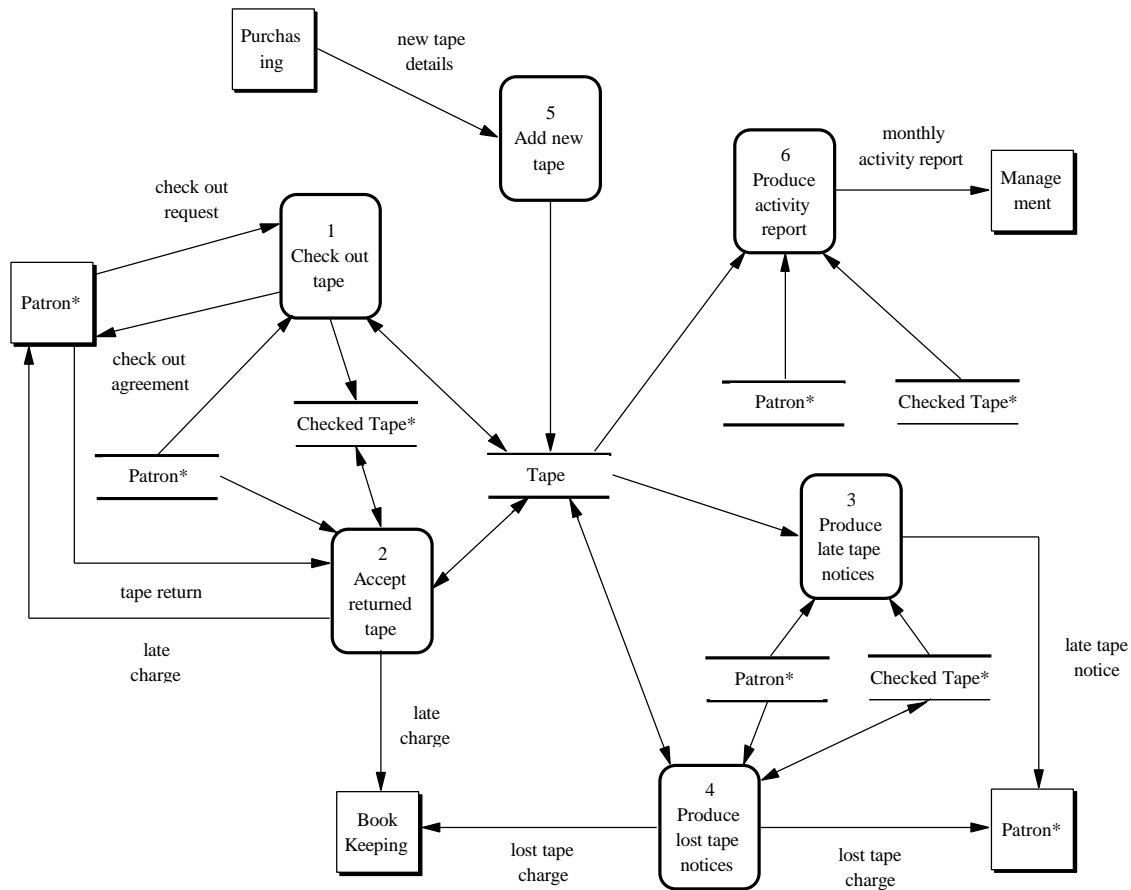
Figure 4: Data Flow Diagram Fragments



First, essential systems analysis focuses on the system's environment as the basis for determining requirements. The purpose of the system and the objectives of the system are defined from an external rather than internal point of view. This is a customer-oriented approach which can be readily understood by users. It would also be useful to use the concepts of events when brainstorming initially about system requirements with end-users, either during the initial investigation or during a joint application development (JAD) meeting. Additionally, the system is seen up front as a dynamic entity that interacts with the environment. All affected stakeholders are identified and the scope of the system is readily seen.

The essential model is also balanced and complete. All information system components are included in the model -- inputs, processes, storage and outputs. The event partitioned system model shows the interaction of all of these components graphically. Information engineering and object-oriented analysis produce more abstract models which de-emphasize inputs and outputs and tend to isolate the system from its environment. The essential model is therefore very useful as a communication tool early in the project.

**Figure 5: Event Partitioned System Model
(Diagram 0)**



The list of events and corresponding essential activities can be used to develop the list of logical processes required early in the information engineering lifecycle. Similarly, the sequence of events can be used to highlight process dependencies that are also modeled in information engineering. Additionally, events can be used to define changes in objects in the system that move an instance of an object through an object lifecycle. The event-response paradigm also anticipates the object-oriented approach to systems, where encapsulated methods are triggered by messages.

CASE support for essential system analysis can be implemented in an entirely different way than has been done for structured analysis. The event table approach organizes system information in a way that could allow the CASE tool to generate graphical models of the system rather than being primarily a drawing tool. For example, the event table can be used to generate the context diagram. It could also generate a set of DFD fragment skeletons which already include external entities, data flows and named processes. Only data stores need to be added to the fragments. Finally, diagram 0 could be generated automatically from the set of DFD fragments.

There are many possibilities for using essential systems analysis techniques in integrated CASE tools. When integrated with an information engineering CASE tool, the event table can generate

the list of logical processes and the DFD fragments can be used to populate the process-entity interaction matrix with read, write and update information. Then, structured English process descriptions can become process action diagrams. Alternatively, the information engineering subject-level data model and the process-entity interaction matrix could be used to complete the DFD fragments.

Strategies for Use in Training

It is argued here that essential systems analysis should replace structured analysis in courses covering analysis and design methods both to increase the rigor of the modeling process and to reduce the amount of time spent on the structured approaches. And then, as alternative approaches are added to the course, essential systems analysis should be taught first to provide both a historical perspective and a solid foundation for system modeling.

When used in the classroom, data flow diagramming per se can be de-emphasized in favor of more fundamental systems concepts such as the importance of user requirements, logical models, graphical system models, functional decomposition, data modeling, following a defined analysis process, and producing concise and non-redundant system documentation. Even as alternative methodologies such as information engineering and object-oriented analysis and design are inevitably introduced in the classroom or required on the job, essential systems analysis gets the analyst thinking more rigorously about information systems and provides a foundation for learning and using these other tools and techniques.

One strategy is to introduce system modeling using essential systems analysis and then take a broader view of systems planning to introduce information engineering. A system example like the one given above is assumed to be part of an integrated whole, so the point can be made about the need for an information system plan prior to embarking on business area analysis projects, particularly where data created by one system are used by another as in the example.

The similarities between information engineering and essential systems analysis for requirements specification can also be emphasized. Both model data, processes and their interactions. Essential systems analysis can also be used to illustrate how a CASE tool can automate the generation of documentation if the right kind of information is stored in the repository. When essential systems analysis has been covered in courses, it has been easy for students to see that a CASE tool should generate drawings for the analyst rather than the other way around. Students are quite disappointed when given a CASE tool that mainly is a drawing tool. The integrated architecture of the information engineering CASE tools becomes easier to understand when students have thought through the possibilities of CASE support.

The essential systems analysis methodology can also be compared to object-oriented analysis. One way to introduce object-oriented analysis is to generate a list of events for a system that mainly keeps records about entities and perhaps generates one periodic report. The list of events becomes quite long because each of the data entities in the system can go through a cycle of being added, modified, viewed, and then eventually deleted. The point can be made that most of the essential activities are custodial.

The object-oriented analysis approach described by Coad and Yourdon (1991) can then be shown to be quite valuable in cases such as this. Methods are assumed for adding, modifying, showing and deleting instances of each object. Therefore, there is no reason to model all of the custodial processes. The relatively simple object-oriented model can be shown to give the same amount of information about system requirements as a long event list and complicated diagram 0. The same point, of course, can be made about information engineering.

Conclusions

This paper described the essential systems analysis methodology and argued that it should be used in training programs and in practice to complement newer methodologies. It is particularly valuable when used early in the systems planning stage of a system development project because it focuses on the environment, de-emphasizes physical aspects of the existing system, provides a heuristic for decomposing the system into processes, and provides a balanced and complete graphical system model useful for communicating requirements. The emphasis on the data model, process model, and interaction between them is compatible with information engineering. Object lifecycle concepts and the event triggering paradigm are compatible with object-oriented perspectives. Essential systems analysis can make a valuable contribution when training future systems analysts and in the system development process.

References

- Coad, P. & Yourdon, E., *Object-Oriented Analysis (2nd Ed.)*, Yourdon Press Prentice-Hall, Englewood Cliff, NJ, 1991.
- Colter, M. , A Comparative Examination of Systems Analysis Techniques, *MIS Quarterly*, March 1984, pp 51-66.
- Couger, J., Evolution of Business Systems Analysis Techniques, *ACM Computing Surveys*, 5:3 (September 1973), pp.167-98.
- DeMarco, T., *Structured Analysis and System Specification*, New York: Yourdon Press, 1978.
- Gane, C. & Sarson, T., *Structured Systems Analysis: Tools and Techniques*. New York: Improved System Technologies, 1977.
- Martin, James, *Information Engineering Volume II*, Prentice Hall, Englewood Cliffs, NJ, 1990....
- McMenamin, S. & Palmer, J., *Essential Systems Analysis*, Yourdon Press Prentice Hall, Englewood Cliffs, NJ, 1984.

Page-Jones, M., *The Practical Guide to Structured Systems Design*, Yourdon Press Prentice Hall, Englewood Cliffs, NJ, 1980.

Page-Jones, M., *The Practical Guide to Structured Systems Design (2nd Ed)*, Yourdon Press Prentice Hall, Englewood Cliffs, NJ, 1988.

Ward, P. & Mellor, S., *Structured Development for Real-Time Systems: Volume 2 Essential Modeling Techniques*, Yourdon Press Prentice Hall, Englewood Cliffs, NJ, 1985.

Yourdon, E., *Modern Structured Analysis*, Yourdon Press Prentice Hall, Englewood Cliffs, NJ, 1989.