On a New Teaching Paradigm For Information Systems

Fran Gustavson[1]

and

Stephen Choolfaian[2]
IS Department, School of Computer Science and Information Systems
Pace University
Pleasantville, NY 10570, USA

Abstract

The Input -> Process -> Output concept has been a basic teaching paradigm of the computer field since its inception. This notion is embedded in the "Systems Concept," in programming and in the teaching process. Since the beginning of the computer field, many things have changed, including improved speed and access to data, faster and larger processors and memories and vastly improved communications and networking capabilities. Because of these changes, it is time for a new paradigm, one that includes current technologic, theoretical, and conceptual approaches. We call this the "Communication Driven Paradigm." This paper starts by describing the evolution of data processing from its beginning to present times, the changes and realities of each stage, and the relevant descriptive system diagrams. It then presents the new "Communication Driven Paradigm" and its diagramming. This paradigm can be used to describe system development using either object oriented or structured systems analysis and design.

Keywords: information systems, process encapsulation, process uncoupling, object orientation, systems paradigm, descriptive information systems paradigm, structured analysis

1. Phase I: The Beginnings

Since the beginning of the "Data Processing/Computer" era, a number of descriptive and conceptual paradigms have evolved, each reflecting the technical and perceptual realities of its time. Each of the then current technologies led professionals in the field to change the methods that would accomplish their design and functional objectives. These new methodologies were converted into ever changing rules that then, falling into the hands of academics, became "paradigms." These paradigms became the underpinnings of the many courses that were developed over time. Although these philosophic underpinnings were often more implicit than explicit, they were the basis upon which specific courses or disciplines were developed .Early in computing history the electrical engineers' notion of the "Black Box" was incorporated into the analysis of information systems. The black box concept held that a good method of analyzing and describing complex activities was to describe them as an input, a black box, and an output. One would not need to be concerned with the details of what went on in the "black box" until a later time. When analyzing the black box, the same process could be used, namely recursively breaking it into more and more black boxes. This came to be understood as the "Systems Concept;" it consisted of three basic elements: Inputs, Processes, and Outputs. This notion of a system was not only true for the Information System, but for virtually any rationally related ongoing set of activities.

[1] Fgustavson@pace.edu
[2] Schoolfaian@pace.edu

Diagrammatically, this was shown as in Figure 1 (Lawlor 1990).

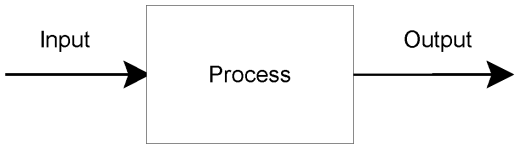


Figure 1. Input/process/output diagram

This early view of a "data processing system" was exemplified by unit record (punched card) equipment. A punched card went into a machine, calculations were made, and a report or modified punched card came out. This could be the entire process, or a step of the process - that is, the black box or one of the lower level black boxes. As the technology advanced, this notion appeared in the physical realities of sequential tape systems. Cards were input, processes occurred, and the results were carried forward on magnetic tapes.

Unit record or early tape systems were often described with "Systems Flow Charts." The diagrams were very much oriented to, and corresponded with, the physical media that carried records from one processing step to another. Systems flow diagrams could easily follow the basic Input, Process, Output (I/P/O) rule, and could also reflect the breaking down, or decomposing, from higher level, more general, black boxes, to lower level, more detailed ones (Lawlor 1990).

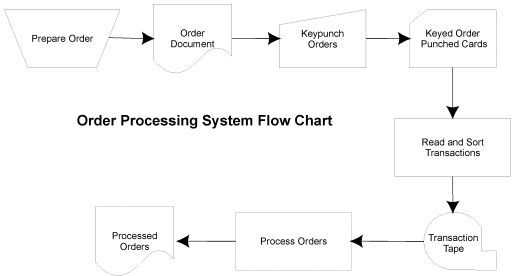
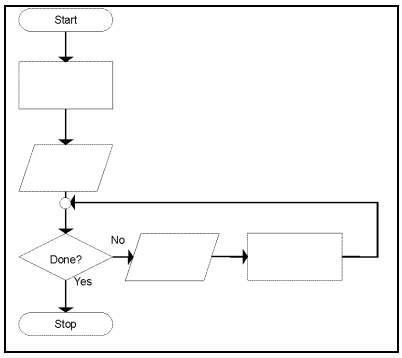


Figure 2. Example of Systems Flow Chart

The I/P/O structure lent itself quite easily to programming. In fact, most languages embodied it. Probably the most often stated "fundamental" of programming was that all programming languages could be described as providing the ability to "read (input)", "assign (process)," "conditionally and/or unconditionally branch," and "write (output)." This was expressed diagrammatically in the "Program Flow Chart" (Gustavson and Sackson 1986) as illustrated in Figure 3.

Figure 3. Example of a program flow chart



Examination of these graphical representations of algorithms clearly shows how the I/P/O notions are applied.

In these early times, processing was transactionally oriented, and while it was understood that files were being used, each typically was processed as a stream of ordered records and was usually processed together with another stream of records ordered in the same sequence. This approach was formalized into the "master file" concept that became the basis of "file based processing.

2. Phase II: File Based Processing

Because of the nature of early storage devices, it was almost inevitable that processing had to be done sequentially to be economic. Direct access was expensive, slow, and of low capacity. Sorting "files" was the most resource consuming part of most applications. (The conventional wisdom of the period was that upwards of 70% of all processing was used to sort.) These realities became imbedded in the "File Based Processing" paradigm. An example of this approach is shown as Figure 4 (Gustavson and Gear 1978).

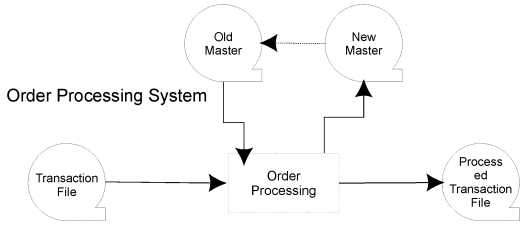


Figure 4. Example of File Based Processing

File based processing typically included the idea of "Old Master to New Master" conversions. At the end of each processing step, the sequential, ordered records, now regarded as files, went through the processing step together. As transactions were matched to master records, changes were made (or not), and a new, updated, master created. Each cycle of the process produced transactional outputs and a new master.

These consistent and common steps introduced the notion of data stores, and therefore the possibility of using the same data in more than one process or application, limited by the sequential nature of the equipment available. It became clear that accessing a specific record, or data element, was quite desirable in some application systems, for example banking and airlines. To meet these needs, a whole new technology was developed. Introduction or improvements of magnetic drums, magnetic disks, mass storage magnetic cards and other such devices enabled economic and rapid direct access to specific data. This ultimately led to the "data driven" Data Base Paradigm.

3. Phase III: "The Data Driven" Data Base Paradigm

Introduction of the Random Access Device as an economically feasible method for the storage and retrieval of data became the basis of "Data Based Systems." This was reflected in the birth of the Data Flow Diagram (DFD). This new concept, and its diagrammatic statement, the DFD, incorporated the idea of independence of data from processes, with the physical and logical notion of the "Data Bank:" a place where data could be stored and accessed by one or more applications within the enterprise. The Data Flow diagram reflected this new reality. Its symbols constituted the world of the application (Whitten, Bentley, Barlow 1994) (Figure 5).

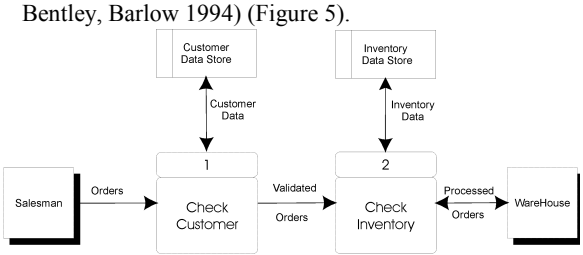


Figure 5. A typical DFD

This new paradigm formed a convenient basis or motivation for most IS curricula. For example, Data Structures and Data Base Management clearly relate to the data aspects of this topography. Systems analysis relates to the identification and specification of information requirements (outputs), data requirements (inputs), and processing requirements and rules. Applications programming relates to the conversion of requirements into procedures and algorithms, and so on.

While not a rigid framework, this approach has been incorporated into similar notions such as Structured Analysis and Design and Structured Programming.

In retrospect, it's fairly obvious that a data driven approach was quite easy to visualize, and often to implement, using a centralized staff together with centralized equipment. The communication of data from outside entities to the processes and then to destinations was quite easily accomplished with, literally, a line, cable or other directly coupled connection. **The reality of data transfer was not far from the diagramming of it.**

Over the years, technology has grown and improved. Processing, data storage, networking and telecommunications capabilities have dramatically increased while costs have substantially decreased. As this occurred, users have increasingly demanded that the benefits of these new technologies be provided now! This has given rise to what we call the "Communication Driven Paradigm."

4. Phase IV: The New "Communication Driven Paradigm."

While centralized "mainframe" equipment dominated the commercial world, large companies typically developed and maintained their own unique systems using this centralized equipment and a corporate staff. The now ubiquitous PC created a new world for the user, hence the analyst, designer, and programmer. The little machine that ***could*** apparently ***did*** provide the ability to do everything, and do it rapidly. As a result, independent designers created applications packages such as Lotus 123, DB3, Access etc., that allowed individual users conveniently to assemble the basic elements of input, output, process and data into simple, relatively easy to use templates or programs, specifically designed to meet that user's "personal computing" needs

At the same time, other independent software developers were also developing "applications packages" for Ordering, Accounting, Payroll and other common business needs. These made good sense for small offices, where one machine typically could handle one or more of these functions with relative ease, or larger settings where single machines could take care of needs in a department or section.

These advances rapidly gave rise to the need to share the data and results obtained at each independent station with other stations requiring the same data, programs, or text. Thus were ushered in LAN's, WAN's, Clients, Servers, and Internets: all of these are tools to enable the sharing of data and programs. In addition to these vital changes to the nature of how systems were constructed and implemented, the decreased cost of processing gave rise to what had previously been a prohibitively costly

method of interacting with users, namely graphical interfaces. The

"windowed," "point and click" technology provided the user a means of conversing with the computer and its processes in what seemed to be a more natural way. This method of interfacing was, whatever its merits and demerits, rapidly institutionalized as the standard user interface. With this new standard user interface, what had earlier been characterized simply as input, still resulted in input being passed to whatever was then receiving it, but the input process required much more processing (for the graphical user interface) than did the earlier, simpler, input.

Thus was born the need for a new paradigm, an enhanced "input, output, data, process" paradigm. As we perceive it, this new paradigm should include:

* User Interface Processes - These "modules", "functions", or however one names them, are the updated expressions of what were called system inputs or system outputs.

* Data Access Processes - Those activities associated with the storage or retrieval of required data elements.

* Data Manipulation Processes - Those activities associated with the calculation or manipulation of data received from User Interfaces and/or Data Processes and forwarding to User Interfaces and/or Data Processes.

* Communications Processes - Those activities associated with the transfer of data between any of the processes listed above.

These processes are shown in symbol form in Figure 6. Note that the communications process symbol should be understood as bi-directional, that is, messages are normally transmitted both to and from the processes that are connected by the symbol.

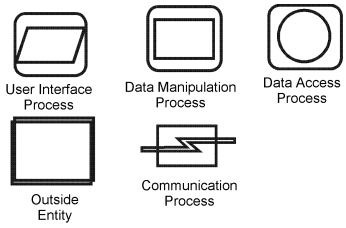


Figure 6. The symbols of the “Communications Driven Paradigm”

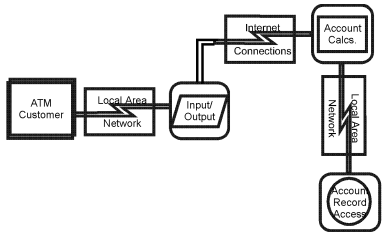
In this new classification scheme, we recognize that **while the reality of data transfer is intuitively close to that expressed by the old diagramming methods, it actually encompasses the processing and transmission of data.** Thus the communication process symbol implies both processing and transmission. Similarly, the process symbol contains an input/output one to represent the processing required to implement the user interface (while maintaining its intuitive “one step flavor”. It further allows the conceptualization of this process to be as an asynchronous, instead of strictly synchronous, one.

Data manipulation and data access have also been enclosed in process symbols, again reflecting the increased processing required to preserve the “one step flavor” of the diagram and their possible asynchronicity.

This classification scheme provides a framework within which all systems can be described. It allows descriptions to be made using existing tools or techniques, while allowing each module to be decomposed into sub modules as required. All systems can be described at almost any level with combinations of these elements. Each element can represent hardware or software, and it is compatible with other parallel concepts such as Object Orientation, Data Distribution, Client/Server and the Internet.

Figure 7 below shows an example of an Automatic Teller Machine system diagrammed using our new symbol set.

Figure 7. A diagram of an ATM machine with multiple associated functions, drawn using out new symbol set.



In addition to providing a way to describe systems, the Communication Driven Paradigm provides an excellent framework for construction of a curriculum. In addition to the core knowledge of the field in areas such as programming, management and communication skills, implementing any of the major Processing Functions will require additional skills and knowledge taught as part of IS/IT curricula. For example, Data Processes would require programming, data structures, data base management and so on. Communications courses might include computer architecture, e.g. buses, channels, LAN's, switching, and so on. User Interface Processes might require skills in using interface development tools such as Visual C++, Java, Javasript, HTML, XML and so on.

5. Conclusion

How we attack problems is often defined by the way we describe them. In today's technologic world, rapid changes in the hardware and software available, and the impact of these changes on our users, has made the teaching about and development of systems a different process than we have had in the past. Our previous view was that of a sequential set of steps that constituted a system or procedure. The new reality is that of a set of independent activities, operating asynchronously. **Our old diagrammatic and structural model cannot accurately depict this new reality. The new "Communication Driven Paradigm," can.**

6. References

Gustavson, Frances G. and C. William Gear, *Applications and Algorithms in Business*, Chicago, Illinois, Science Research Associates, Inc., 1978

Gustavson, Frances G. and Marian V. Sackson, *Problem Solving and Basic: A Modular Approach, Second Edition*, Chicago, Illinois, Science Research Associates, Inc., 1986

Lawlor, Steven C., *Computer Information Systems*, San Diego, California, Harcourt Brace Jovanovich, Publishers, 1990

Whitten, Jeffrey L., Lonnie D. Bentley, Victor M. Barlow, *Systems Analysis and Design Methods*, Boston, Massachusetts, Richard D. Irwin, Inc. 1994