Vincent A. Mastro
TECHNOLOGY INFORMATION PRODUCTS CORPORATION
12 New England Executive Park
Burlington, MA 01803
(617) 273-5818

Vincent Mastro is currently a Consultant for Technology Information Products Corporation. He is involved in the teaching and consulting of their technologies, which through analysis, design and implementation develop more effective information systems.

THREE DIMENSIONAL SYSTEM DEVELOPMENT

The primary goal of the software engineering community continues to be the development of economical information systems that are user-friendly, totally integrated, completely documented, easily maintained and, most importantly, support the business. In response, the software engineering community has borrowed many techniques from the "physical" engineering disciplines. Although valuable, these techniques have not proven to be entirely successful.

One technique used by the "physical" engineering community that has not been adopted is based on a 3-dimensional diagramming technique that documents the current level of understanding while providing the foundation for communication and analysis. This technique diagrams the top, front and side views of the products designed by the physical engineers.

The use of a "3-dimensional" technique by the software world requires first, the identification of a software version of the three "views", then the use of a system development methodology that supports the modeling of those views. The 3-Dimensional System Development approach will provide the software engineering community with a comprehensive method of identifying and communicating the requirements of almost any business and should result in the production of more effective information systems.

VINCENT A. MASTRO June 7, 1985

THREE DIMENSIONAL SYSTEM DEVELOPMENT

Will information systems that completely fulfill the needs of the user community ever be developed? Are information systems that are truly user friendly, support the business, totally integrated, completely documented and easily maintained nothing more than a dream? Obviously, the answer to these questions and the development of such an information system depends on many factors including, but certainly not limited to, system development methodologies, management commitment, resource availability and the skills of both the user community and the development team.

These factors have become collectively known as software engineering, a discipline which has been growing steadily since the early seventies. Software engineering can be defined as the use of reproducible methods and principles that result in economical software that is reliable and performs the functions for which it was intended. In an effort to become a discipline that conforms to the above definition, the software community has borrowed many techniques from the "physical" engineering disciplines. These techniques include: identification of a software life-cycle, enforcement of project management principles, adherence to structured coding languages and design processes, use of configuration management or change control procedures and the documentation of every step in the process. The software community has, given the infancy of their craft, done a reasonably good job - but have they borrowed enough of the "physical" engineering techniques?

An investigation into the similarities between the software and the "physical" engineering disciplines may help identify how the "physical" engineering disciplines accomplish similar tasks. This investigation must begin with a very simplistic description of what the "physical" engineering disciplines do. Their process starts with the identification of a need, usually as a result of a perceived problem. The problem, which usually relates to a physical structure, is studied so that it can be understood, recognized as "real" and communicated to all concerned. This is accomplished via a simple diagramming technique (Figure 1) that converts the problem into a set of diagrams that:

- o Document the current level of understanding
- o Provide the mechanism for communicating that level of understanding

Once documented, communicated and understood, the problem exists as a documentation package in the form of diagrams, models and supporting text. This "As Is" model is then used to form the baseline from which analysis is performed and a variety of approaches to solving the problem are developed. Eventually, one of the approaches is chosen (referred to as the "To Be" model) and is given to manufacturing where it is further refined and eventually assembled into the intended product.

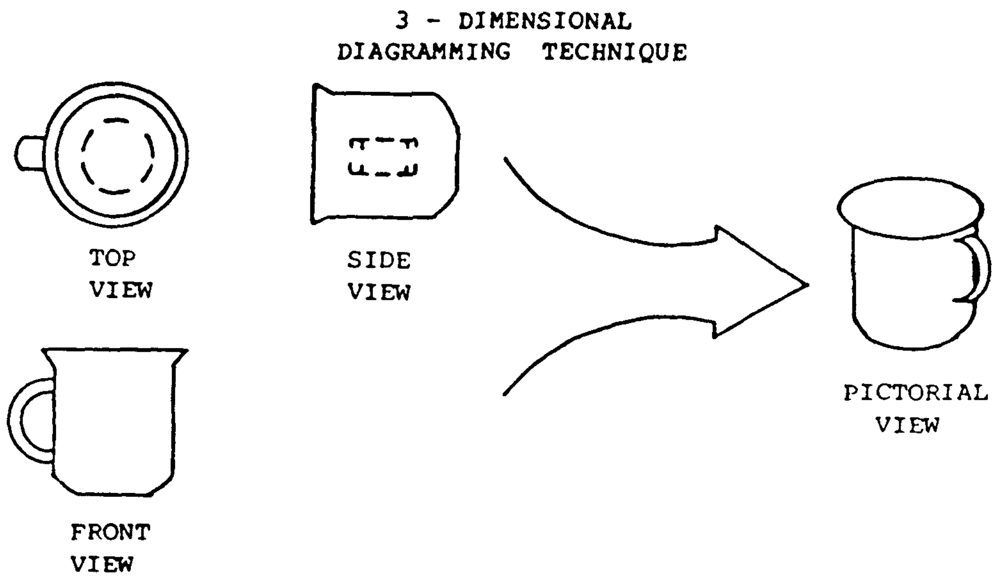


Figure 1

The process used by the "physical" engineering community sounds (at this simplistic level) remarkably similar to the software engineering process. The primary difference however, is that the "physical" engineers exist in the physical world and must therefore convert their design into a solid product via some manufacturing process. Specifically, the "physical" engineering disciplines communicate via 3-dimensional diagrams as illustrated in Figure 1. These diagrams or blueprints are continuously refined and transformed into the final product. As a result, the diagrams become the primary communication vehicle used by everyone involved in the project. The software engineering community does not. Their final product is not physical, it is another kind of communication - communication from man to machine i.e., code. If the software engineers are to adopt the communication technique used by the "physical" engineers, they must first determine exactly how the "physical" engineers use the diagrams as communication vehicles and attempt to adapt the diagrams to their needs. The goal of the software engineering community then, is to develop a software engineering version of the 3-dimensional diagramming technique so that they can enhance communication between management, users, analysts, coders and the computer.

"PHYSICAL" ENGINEERING VIEWS The basis for diagramming the "front", "top" and "side" views of the cup in Figure 1 is provided by the natural relationship between the height, width and depth of all objects within the physical world. The ability of the "physical" engineering community to project this natural phenomenon into a diagramming technique results in an increase in their knowledge about the item under study. This increased knowledge is directly attributable to the synergistic effect of the diagramming technique. For instance, each of the views illustrated in Figure 1, when taken independently, do not have enough information to fully describe the cup. However, when the views are reviewed together, they provide more information than the pictorial view alone thereby enhancing the level of understanding.

These views also verify one another. For instance, except for the handle, the front and side views of the cup are identical. If they were not, either it is not a cup or it has been drawn inaccurately. This natural verification process also plays a key role in the integration of the three views. Specifically, the three views are all integrated together into one diagram as indicated by the pictorial view of the cup (Figure 1). It would be impossible for the views to be integrated into one picture if their measurements were inconsistent and/or inaccurate.

Another important factor associated with these views is that each view is completely described. This is illustrated by the side view of Figure 1. The dashed rectangle represents the size and location of the cup's handle even though it cannot be directly seen (it resides behind the viewing plane).

The "physical" engineering diagramming technique is also scalable. It can be used to indicate varying levels of detail and/or completeness. Scalability helps establish the boundaries and/or scope of the diagram. Figure 2 is an example of scalability . It illustrates how this diagramming technique can be used to define an entire system, a sub-system or a piece-part.

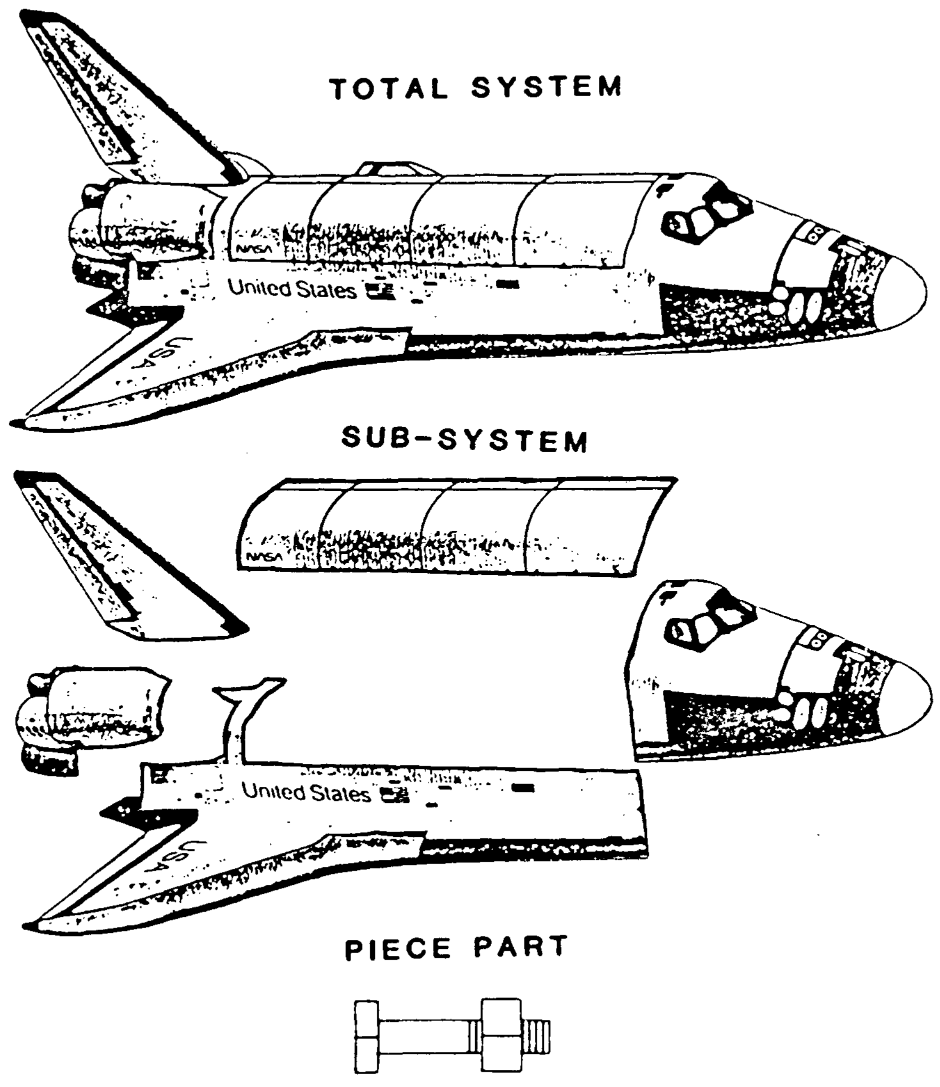


Figure 2

SOFTWARE ENGINEERING VIEWS

Is there a software engineering version of the front, top and side views which can be used as the foundation for modeling a business and its associated information systems? Information is composed of data. The data which makes up information, has natural relationships and/or associations to other data. Many of these natural relationships are established by the business environment in which they exist. These relationships in turn, are defined as a result of the data's usage within the context of those business functions. Given this level of understanding, the natural basis for the three software engineering views would therefore be:

- o Functional - defines the business functions to be performed by the information system
- o Information Usage - defines the information used by those business functions
- o Data - defines the natural associations (relationships) among the data required by the business

Based on this very brief analysis and the previous discussion concerning software engineering in general, the components of a comprehensive system development methodology are:

- o Economical
- o Reproducible
- o Picture driven
- o Model the business from two perspectives
  - As is
  - To be
- o Solve the intended problem
- o Maintainable

The pictorial component must be:

- o Scalable
- o Easily drawn
- o Understandable to all concerned
- o Completely describe each view
  - Functional
  - Information Usage
  - Data
- o Provide the means for clear communication from problem identification through to code development

Each view must be:

- o Integrative and consistent with the other views
- o Verifiable by the other views

Table I relates the three software views with examples of some of the diagramming techniques currently in use today. The vast majority of system development methodologies incorporate diagramming techniques that model the business using only one or two views. Unfortunately, modeling one or two views with one methodology usually will not allow for the verification or integration of the other view(s) diagrammed using a different methodology. In other words, the diagrams would not be integrative and could therefore not be combined into one diagram as exemplified by the pictorial view of Figure 1. Consequently, the engineer's overall understanding would not be increased because it would be difficult to take advantage of the synergistic effect made possible by the natural associations between the three software engineering views.

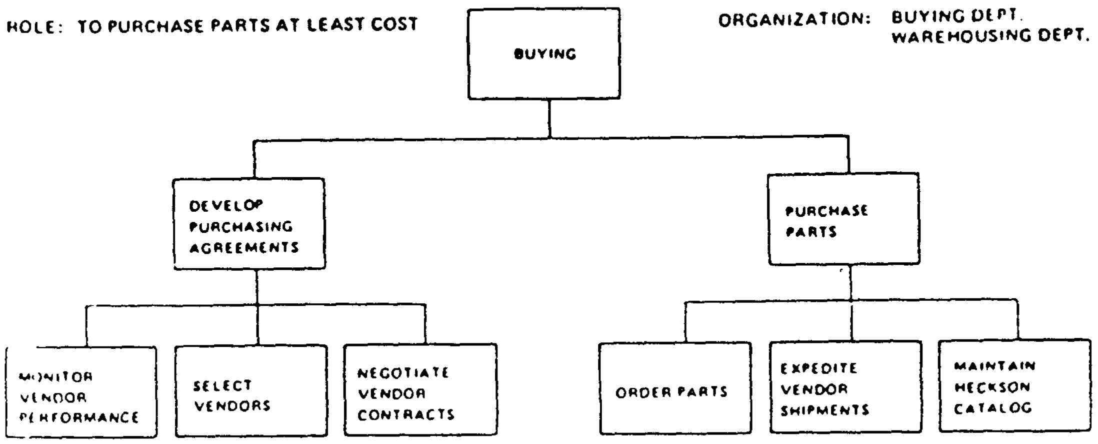
TABLE I

VIEW	TECHNIQUES	EXAMPLES
FUNCTIONAL	HIERARCHIES	FUNCTIONAL DECOMPOSITION NODE TREES
INFORMATION USAGE	NETWORKS	INFORMATION USAGE MODELS DATA FLOW DIAGRAMS
DATA	TABLES HIERARCHIES	RELATIONAL MODELING ENTITY MODELING BUBBLE CHARTS

FUNCTIONAL VIEW

The diagram illustrated in Figure 3 is a functional decomposition of the purchasing needs of a hypothetical company. It is called the "Buying Business" because the term "Buying" is used within the company to describe what they do. As a functional decomposition diagramming technique, it identifies the business functions and their logical relationship to each other. This diagramming technique, called Functional Business Modeling, clearly identifies the beginning and ending functions of the business being modeled thereby establishing its boundaries. A business is defined as a functional component of an enterprise that provides a specific product or service to a specific client or market. As such, a business can be defined as an enterprise, a company within an enterprise or a department in the enterprise's smallest company: hence, it is scalable. A function is defined as a group of activities that describe what must be done on a continuous basis in order to provide a specific product or service. An activity, on the other hand, is a unit of work that describes exactly how to perform a function and must therefore include the identification of conditional, repetitive, inclusive, exclusive, parallel and termination work units. The Functional Business Model then, is a high level diagramming technique which models the functional view.

FUNCTIONAL BUSINESS MODEL


ROLE: TO PURCHASE PARTS AT LEAST COST
BUYING
ORGANIZATION: BUYING DEPT. WAREHOUSING DEPT.
DEVELOP PURCHASING AGREEMENTS
PURCHASE PARTS
MONITOR VENDOR PERFORMANCE
SELECT VENDORS
NEGOTIATE VENDOR CONTRACTS
ORDER PARTS
EXPEDITE VENDOR SHIPMENTS
MAINTAIN HECKSON CATALOG

BUYING BUSINESS

Figure 3

INFORMATION USAGE VIEW Figure 4 contains two diagrams that describe the usage of information by the lowest level functions in the Buying Functional Business Model (Figure 3). Information can be data, reports, forms, intangibles such as knowledge, or tangibles such as parts. These models, appropriately called Information Usage Models, are based on a network diagramming technique that identifies what information is used by the functions on a Functional Business Model. Information Usage Models also highlight the flow of information to or from passive stores of information (i.e., files, warehouse bins or clipboards), external interfaces (i.e., vendors or areas outside the scope of the study), and functions on a different Functional Business Model. Since the Information Usage Models are based primarily on the Functional Business Model, it can be used to verify its accuracy. For example, the functions on both models must be the same, otherwise the Information Usage Model would not accurately describe the usage of information by that business. The verification process is based on a variety of specific analysis guidelines which relate to the use of and/or the sharing of information by those functions. The guidelines would determine if the model was diagrammed accurately. If not, changes to the Information Usage Model would eventually be reflected in the Functional Business Model.

INFORMATION USAGE MODELS

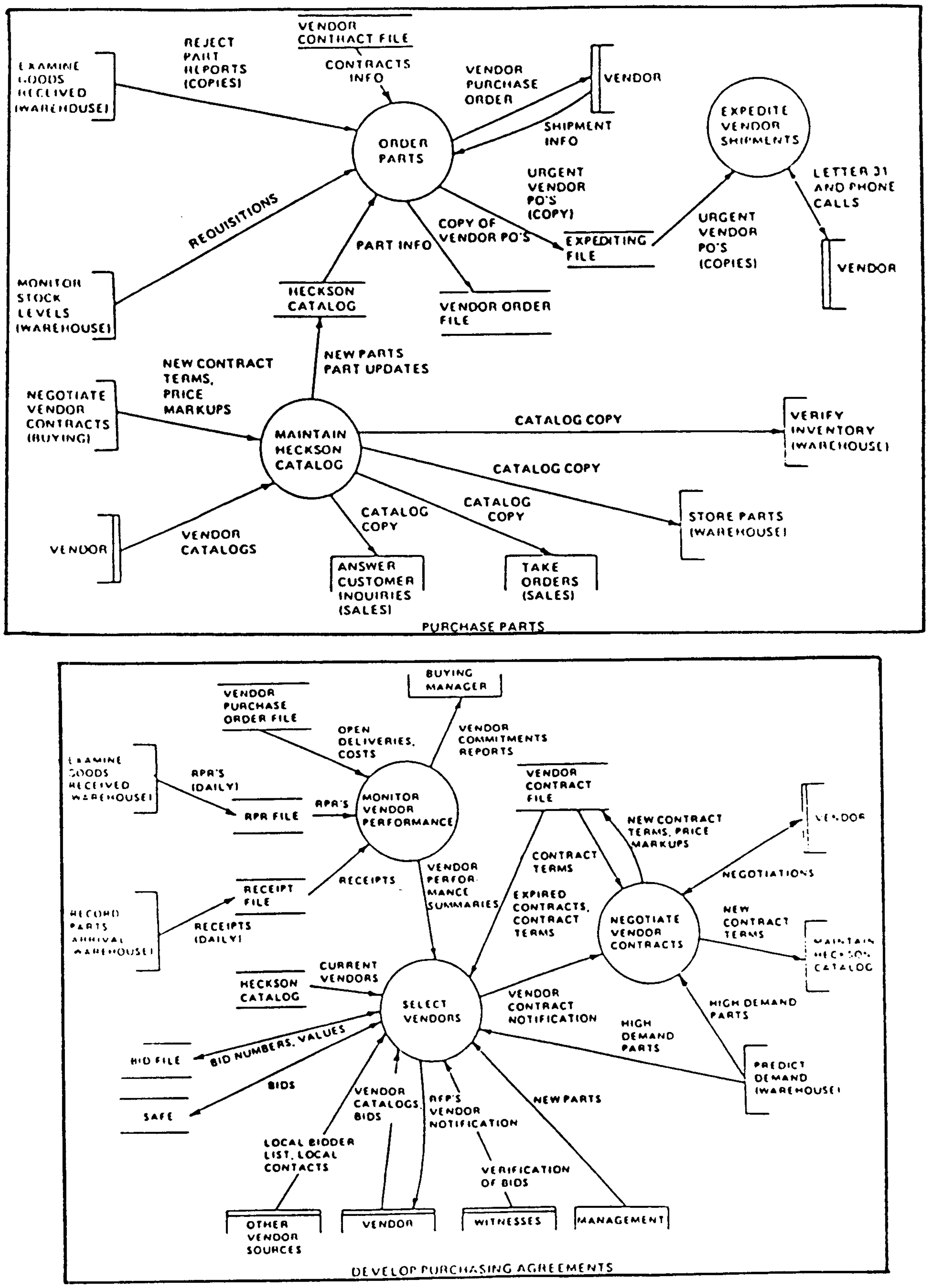


Figure 4

Given a brief description of the notation, Information Usage Models are easily drawn and understood. Since the technique is scalable, Information Usage Models can completely describe the use of information at any level on the Functional Business Model (the lowest level is exemplified in Example 3).

DATA VIEW The data view defines the natural associations and/or relationships among all the data required by a business. An Data Relationship Model is a high level diagramming technique which models the data view. The diagram in Figure 5 is an Data Relationship Model that documents the associations and/or relationships between all of the entities required by the Buying Business.

ENTITY DATA MODEL

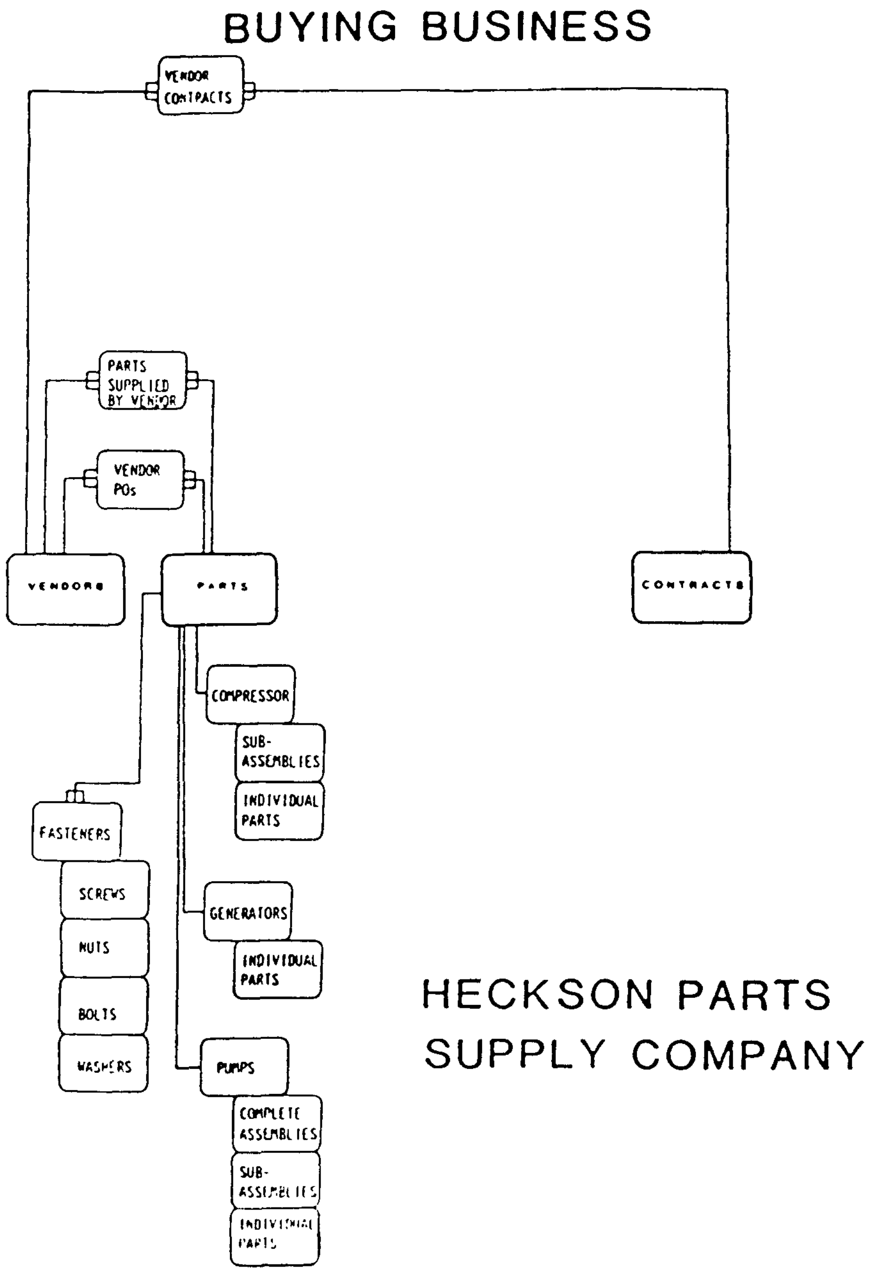


Figure 5

Entities may also be thought of as business resources. They are defined as persons, places, or things that are vital to the business and about which data is or should be stored. The data (also referred to as "properties") describe the entities. The Parts entity for instance, has (at a minimum) the following properties:

- Part Number
- Part Name
- Part Description

The associations and/or relationships are natural connections between entities. For instance, since line items identify the parts that are being ordered by a purchase order, there is a naturally occurring association between a purchase order and line items. Therefore, the natural association exists because a purchase order would not be a purchase order if it did not have line items. In the previous example, one company may allow multiple line items for each purchase order while another company mandates that there be one and only one line item for each purchase order. The management practices of the two companies establish things called "business rules" which define the specifics of the naturally occurring association. For instance, in the former company, the business rule dictates that there is a "one to many" relationship between Vendor Purchase Orders and line items. In the latter company, the business rule dictates that there is a "one to one" relationship.

As the business operates on a daily basis, "events" occur which cause the business to alter data. Expanding upon the previous example, when the inventory levels are depleted to their reorder point, an event occurs which causes the creation of a Vendor Purchase Order. In order for the Vendor Purchase Order to be created, the business must have both the vendor and part information available for use. Events therefore, trigger activities and/or decisions within business functions and cause data to be required and/or altered.

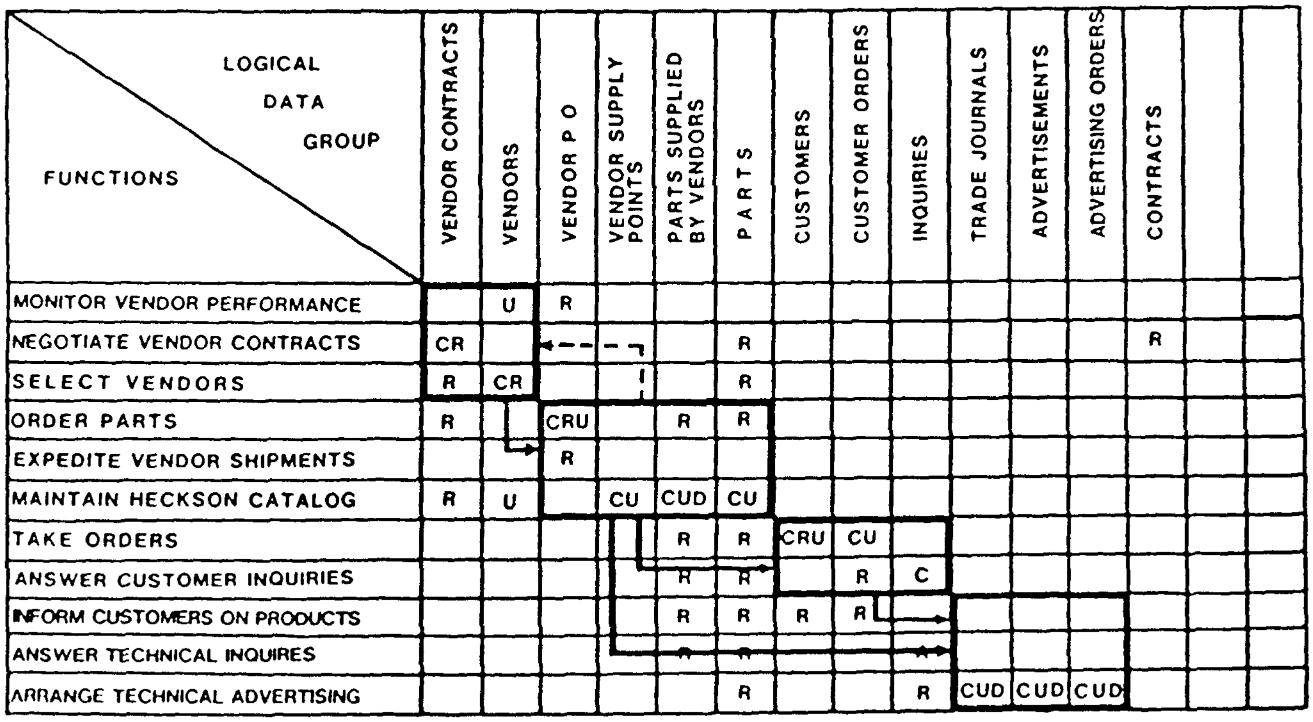
The Data Relationship Model in Figure 5 is a diagram of the data requirements for the Buying Business. In this example, the function, Order Parts, is composed of the activity, Create Vendor Purchase Order. Whenever parts or supplies must be ordered an event occurs which results in the creation of a Vendor Purchase Order. The event which triggers the decision to create the Vendor Purchase Order, is that the reorder point for certain parts was reached. The Vendor Purchase Order is composed of data from the Vendors entity and the Parts entity. The Vendor Purchase Order box in the Data Relationship Model (Figure V) documents the results of the event and is called an intersecting entity. The rules of the business state that a Vendor may have many Vendor Purchase Orders and each Vendor Purchase Order refers to only one Vendor. The diagram also indicates that a Vendor Purchase Order may have many Parts and that each Part may appear on many Vendor Purchase Orders.

In summary, the Data Relationship Model is a diagramming technique which clearly describes the data view from a high level modeling perspective. The Functional Business Model and the Information Usage Model aid in the identification of entities and events. The Data Relationship Model verifies the accuracy of the Functional Business Model(s) and the Information Usage Model(s). It is easily understood, scalable and it can describe the data and its association to other data required by the entire business or any portion there of.

INTEGRATED VIEW

Finally, the three views are integrated together via the Function/Logical Data Group Matrix modeled in Figure 6. The functions on the Function/Logical Data Group Matrix are extracted directly from the Functional Business Model. The Logical Data Groups are extracted from the Data Relationship Model. Since Logical Data Groups are a collection of properties which describe an entity, there is a minimum of one data group for each entity* on the Data Relationship Model. The third view, information usage, is identified by the entries in the intersection between function and data group. These entries are called "usage patterns" and identify how the functions use the information in the data groups. Specifically, a function may either Create, Read, Update, Delete or Archive information to data groups. The result is a diagramming technique which provides the software engineer with a synergistic understanding, because it models the Functional, Information Usage and Data views simultaneously.

FUNCTION/LOGICAL DATA GROUP MATRIX
WITH INFORMATION FLOWS
(SALES AND BUYING)



FUNCTIONS \ LOGICAL DATA GROUP	VENDOR CONTRACTS	VENDORS	VENDOR P O	VENDOR SUPPLY POINTS	PARTS SUPPLIED BY VENDORS	PARTS	CUSTOMERS	CUSTOMER ORDERS	INQUIRIES	TRADE JOURNALS	ADVERTISEMENTS	ADVERTISING ORDERS	CONTRACTS		
MONITOR VENDOR PERFORMANCE		U	R												
NEGOTIATE VENDOR CONTRACTS	CR					R							R		
SELECT VENDORS	R	CR				R									
ORDER PARTS	R		CRU		R	R									
EXPEDITE VENDOR SHIPMENTS			R												
MAINTAIN HECKSON CATALOG	R	U		CU	CUD	CU									
TAKE ORDERS					R	R	CRU	CU							
ANSWER CUSTOMER INQUIRIES					R	R		R	C						
INFORM CUSTOMERS ON PRODUCTS					R	R	R	R							
ANSWER TECHNICAL INQUIRES					R	R			R						
ARRANGE TECHNICAL ADVERTISING						R			R	CUD	CUD	CUD			

C - CREATES A - ADDS R - READS U - UPDATES D - DELETES

Figure 6

* Detail level modeling techniques group the properties within an entity according to more stringent rules called normalization. This often results in the original entity consisting of multiple Logical Data Groups.

The development of the Function/Logical Data Group Matrix as well as the analysis of it, verifies the accuracy of the other three diagrams. The verification process often leads to changes in the Data Relationship Model, Information Usage Model and the Functional Business Model. The resulting changes produce a more accurate description of the business being modeled thereby increasing the Software Engineer's overall knowledge. The Function/Logical Data Group Matrix may be used to identify applications and to establish the sequence for application development based on the data requirements of the business.

MODELING TECHNIQUES The models shown in Figures 3-6 are high level modeling techniques. As a result, they do not model all of the requirements of a particular information system. However, they do provide a clear understanding of the business problem so that an approach to resolving it may be planned. These planning level models also provide a firm foundation for the development of more detailed models in much the same way the "physical" engineering community uses their high level models to produce precise mathematical formulations, measurements and scale models of their diagrams. Table II identifies both the high level and detail level diagramming techniques used by the system development methodology previously discussed.

TABLE II

VIEW	HIGH LEVEL DIAGRAM	DETAIL DIAGRAM
FUNCTIONAL	FUNCTIONAL BUSINESS MODELS	DETAIL ACTIVITY MODELS
INFORMATION USAGE	INFORMATION USAGE MODELS	DATA USAGE DIAGRAMS
DATA	DATA RELATIONSHIP MODELS	RELATIONAL DATA MODELS

The system development methodology illustrated in Figures 3 - 6 uses an integrated "3-dimensional" approach and leads to a more comprehensive understanding of the requirements. This is accomplished primarily because of the emphasis on pictures rather than text. The result is the development of integrated information systems that fulfill the needs of both the user and data processing.

SUMMARY The primary goal of the software engineering community continues to be the development of economical information systems that are user-friendly, totally integrated, completely documented, easily maintained and, most importantly, support the business. In response, the software engineering community has borrowed many techniques from the "physical" engineering disciplines. Although valuable, these techniques have not proven to be entirely successful.

One technique used by the "physical" engineering community that has not yet been adopted by the software engineering community is based on a 3-dimensional diagramming technique that documents the current level of understanding while providing the foundation for communication and analysis. This technique diagrams the top, front and side views of the products designed by the physical engineers.

The use of a "3-dimensional" technique by the software world requires first, the identification of a software version of the three "views", then the use of a system development methodology that supports the modeling of those views. The 3-Dimensional System Development approach will provide the software engineering community with a comprehensive method of identifying and communicating the requirements of almost any business, resulting in the production of more effective information systems.

BIBLIOGRAPHY

Exxon Corporation, "Structured Systems Analysis(SSA) Reference Manual." Technology Information Products Corporation, Oct. 1982.

Jacobsohn, Herb, "Readers' Forum - Why Have Info Systems Failed?" Datamation, Aug. 15, 1984, 147.

Mendes, Kathleen, "Structured Systems Analysis: A Technique to Define Business Requirements." Sloan Management Review, Massachusetts Institute of Technology. 21,4 (1980), 51-63.

Popek, A.B. "A Guide to Business Information Planning, Reference Manual." Technology Information Products Corporation, June, 1984.