OBJECT ORIENTED DATA MODELING: ENTITY LIFE HISTORIES

David C. Hay
Essential Strategies, Inc.

INTRODUCTION

This paper presents a technique for tightly coupling models of business processes to those of data structure. It is largely based on the work of Chris Bird, John Hall, and Keith Robinson of Model Systems Consultants, Inc.,[1] although the technique was originally described by Michael Jackson (no, not the popular singer) in his 1983 book, *System Development*,[2] and it is incorporated into the SSADM methodology widely used in Europe.[3]

Oracle's CASE*Method represents a particularly sophisticated approach to entity/relationship modeling, producing models which are both rigorous and accessible to users. The resulting models form a very good foundation for data base design.

The method is weaker, however, when it comes to capturing the essence of a company's business functions rigorously enough and in enough detail to form a solid basis for developing software. The function side of analysis is explicitly separated from the data side, and the function hierarchy remains a somewhat subjective kind of model. The "CRUD" matrix. showing which functions create, retrieve, update, and delete which entities, provides some guidance for the way programs will use data, but without more rigor in the definition of functions, even this remains a weak tool.

In the 1970's Ed Yourdon's and Larry Constantine's principles of structured design[4] provided real guidance in defining modules, with rules for defining each module's use of data. Unfortunately, however, the rules were defined in terms of third-generation programming languages, and don't apply in the same way to fourth-generation languages and forms tools such as SQL*Forms.

The move towards an object-oriented approach to programming, however, and by extension to design and systems analysis, has provided some insights into modeling a business. Where CASE*Method represents data and function modeling as two parallel operations, object-orientation shows them as much more integrated, bringing together not only the function hierarchy and the data model, but other modeling techniques as well.

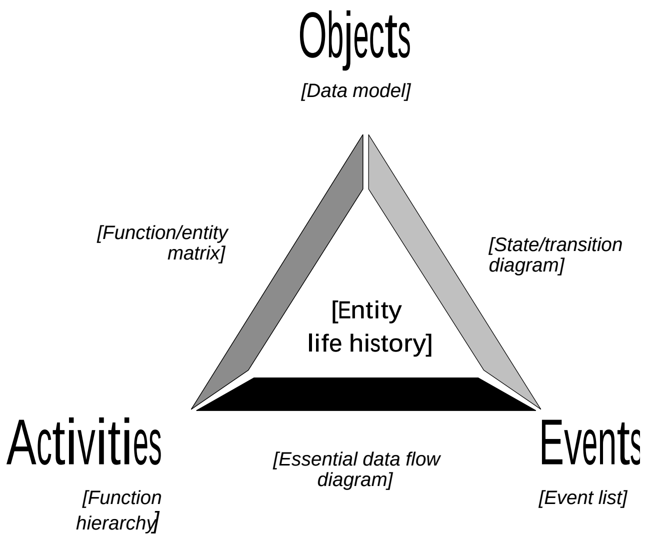

Figure 1: Models

Figure 1 shows the three aspects of an organization that are the subject of most modeling techniques in use today, and how the techniques are related to them. The three aspects are:

— The *objects* that the business manipulates: the things of significance about which the organization must hold and maintain information.

— The *functions* or *activities* performed by the business.

— The *events* in the world that cause those functions to be performed.

The *data model (entity/relationship model)* provides a picture of the organization's objects. (An "object" is an occurrence of an entity, and an "object class" is an entity.) The *function hierarchy* describes its functions. *Events* are not so much modeled as listed, but they play an important part in the other models.

The relationships between functions and objects are usually represented in *function/entity matrices ("CRUD" matrices)*, showing which functions create, retrieve, update, and delete which entities. Relationships between events and functions are shown effectively in *"essential data flow diagrams"*, as described originally by McMenamin and Palmer.[5] The link between events and objects is shown by *state/transition diagrams,* described in, among other works, *CASE Method: Function and Process Models*, by Barbara and Richard Barker and Cliff Longman.[6] This shows the sequence of states an entity goes through as it responds to events.

In the middle of the triangle is *entity life histories*, the subject of this paper. These models show the events that affect each entity and the operations (functions, activities) that are triggered as a result.

THE DATA MODEL

Figure 2 shows a portion of a data model for a purchasing and maintenance environment. The model describes purchase orders used to acquire materials and the work orders which consume them. The letters in parentheses next to the relationship names describe the referential integrity rules: "C" designates the "cascade" rule that if an occurrence of a parent entity is deleted, all children are to be deleted. "R" is the "restricted" rule that a parent may not be deleted if children exist.

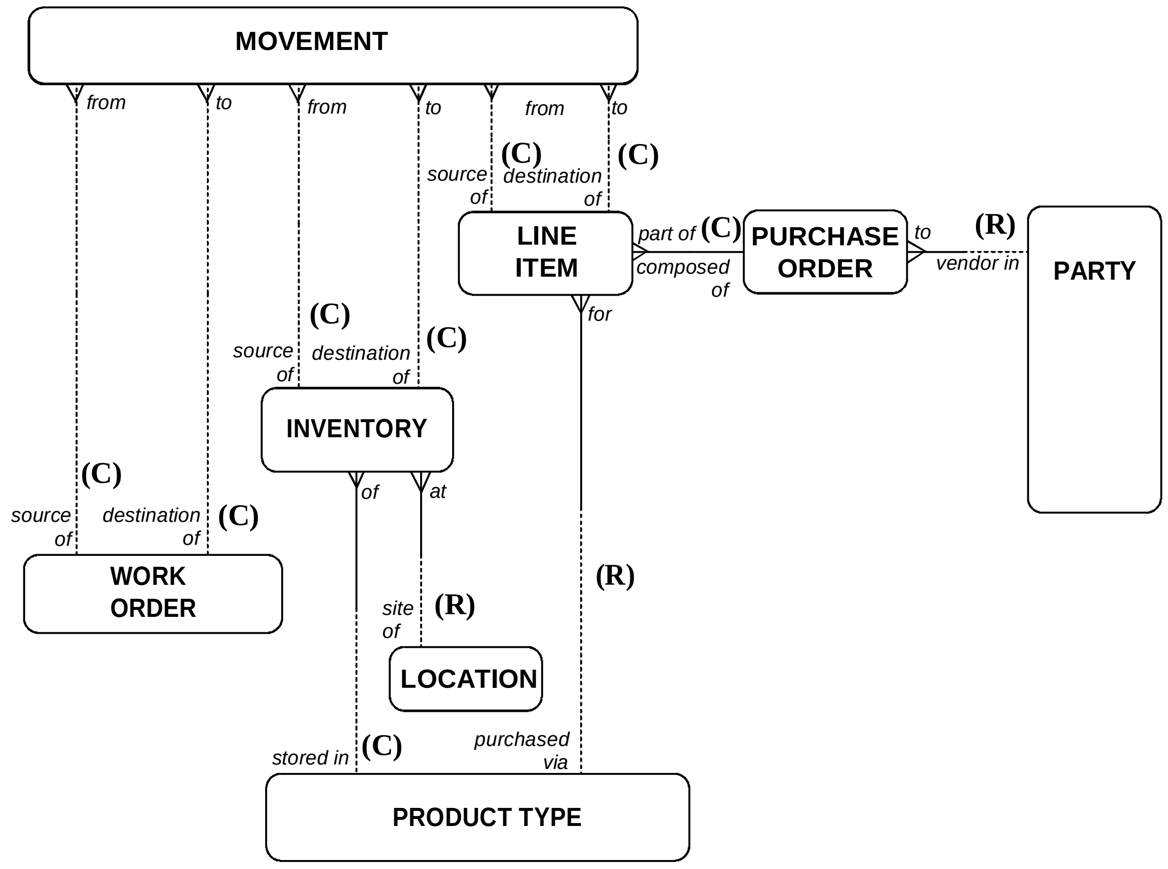

Figure 2: Entity/Relationship Model

The central entity in this model is PURCHASE ORDER, which is an agreement to pay for specified PRODUCT TYPES which are to be supplied by a PARTY. A PURCHASE ORDER is itemized by LINE ITEM, each of which is the fact that a particular PRODUCT TYPE has been specified in a particular PURCHASE ORDER.

An INVENTORY is the fact that a LOCATION may be used to store a PRODUCT TYPE. (A LOCATION may store more than one, and a PRODUCT TYPE may be stored in more than one LOCATION.)

The entity representing transactions that move material in, out, and through the plant is MOVEMENT. When a quantity of PRODUCT TYPE is received against a PURCHASE ORDER, for example, a MOVEMENT is recorded *from* the LINE ITEM where it was specified *to* a particular LOCATION. Should the PRODUCT TYPE be found to be faulty, it may be returned via a MOVEMENT *from* the LOCATION *to* the LINE ITEM (and by implication, to the vendor PARTY).

When the time comes to use the PRODUCT TYPE, a work order is prepared authorizing that use, and material is drawn *from* an INVENTORY *to* the WORK ORDER, via a MOVEMENT transaction. If not all the issued material is used, another MOVEMENT transaction can reverse the process.

Sometimes it is necessary to move material *from* one INVENTORY (LOCATION) *to* another, so another kind of MOVEMENT records this. Sometimes also, physical counts of the INVENTORY are done, and adjustments increase or reduce the balance. An increase can be recorded as a MOVEMENT *to* an INVENTORY with no "from" indicated, and a decrease can be recorded as a MOVEMENT *from* an inventory with no "to" identified.

Attributes of some of these entities are shown in Table 1. The attributes are described here in "column" terms, including foreign and primary keys. Each attribute is shown to be fixed, replaceable, a foreign key, or part of the primary key. This determines, among other things, whether the attribute can be changed once it is set. An activity or operation may change the value of a replaceable attribute, but it could never change the value of a primary key or a fixed attribute.

	Entity	*Attribute*	*Attribute Type*
1	*Inventory*	*Model number*	*| Compound*
		Location ID	*| primary key*
		Quantity	*Replaceable attribute*
2.	*Line Item*	*PO number*	*| Compound*
		Line number	*| primary key*
		Model number	*Foreign key*
		Quantity	*Replaceable Attribute*
		Price	*Replaceable attribute*
		Total received	*Calculable attribute*

3.	*Location*	*Location ID*	*Primary key*
		Description	*(Fixed? Replaceable) attribute*
4.	*Movement*	*Movement ID*	*Primary key*
		Movement type ID	*Foreign key*
		From LINE ITEM	*Foreign key*
		From INVENTORY	*Foreign key*
		From WORK ORDER	*Foreign key*
		To LINE ITEM	*Foreign key*
		To INVENTORY	*Foreign key*
		To WORK ORDER	*Foreign key*
		Movement date	*Fixed attribute*
		Quantity	*Fixed attribute*
5.	*Product type*	*Model number*	*Primary key*
		Description	*Replaceable attribute*
6.	*Purchase order*	*PO number*	*Primary key*
		PARTY (vendor) ID	*Foreign key*
		Due date	*Replaceable attribute*
		Terms	*Replaceable attribute*
		Issue date	*Fixed attribute*
		Closure date	Fixed attribute
		Cancel date	*Fixed attribute*
		Status	*Replaceable attribute*

Table 1: Attributes

For purposes of this case study, the entities shown in Table 1 are subject to the events shown in Table 2.

	Event
1.	*Product type definition*
2.	*Product type description change*
3.	*Product type discontinuance*
4.	*Purchase order issuance*
5.	*Purchase order change*
6.	*Purchase order close*
7.	*Purchase order deletion*
8.	*Location definition*
9.	*Inventory definition*
10.	*Inventory discontinuance*
11.	*Location discontinuance*
12.	*Receipt of goods*
13.	*Return to vendor*
14.	*Transfer*
15.	*Adjustment of inventory quantity*

Table 2: Events

McMenamin and Palmer, in their *Essential Systems Analysis*[7] describe an essential activity as the complete response to an event. Each of the events listed above, therefore, corresponds to an essential activity, or in CASE*Method terminology, an "elementary business function". The function/entity matrix, then, can be modified to be a "event/entity" matrix. For example, this is shown in Table 3.

A few things should be noted about this matrix. First, where a relationship is "restricted", the deletion of one entity "reads" the controlling entity. For example, the event "Product type discontinuance" reads the entity LINE ITEM, to see if it can be done. Similarly, where a relationship "cascades", the deletion of the parent entity also deletes all the children. For example discontinuing a PRODUCT TYPE also deletes all related occurrences of MOVEMENT.

Also, whenever a child is created (for example, INVENTORY), the parent entities are "read" (in this case, LINE ITEM and PRODUCT TYPE).

		INVENTORY	LINE ITEM	LOCATION	MOVEMENT	PRODUCT TYPE	PURCHASE ORDER	WORK ORDER
1.	Product type definition					C		
2.	Product type description change						U	
3.	Product type discontinuance	D	R			D		
4.	Purchase order issuance			C			R	C
5.	Purchase order change		U			R	U	
6.	Purchase order close						U	
7.	Purchase order deletion			D				D
8.	Location definition			C				
9.	Inventory definition	C	R			R		
10.	Inventory discontinuance		D			D		
11.	Location discontinuance		R		D			
12.	Receipt of goods	U	U		C		U	
13.	Return to vendor	U	U		C		U	
14.	Transfer	U*			C			
15.	Adjustment	U			C			

Table 3: Entity/Event Matrix

***** Note that two occurrences of INVENTORY are affected — the "from" INVENTORY and the "to" inventory.

MOVEMENT

Beginning with the top of the data model, Figure 3 shows the entity life history for MOVEMENT. Each event is represented by a box. the entity whose history it is is shown by a round-cornered rectangle at the top.

Note that since MOVEMENT is only a transaction event, occurrences are normally only created. this "Creation" may be either as a "Receipt of goods", "Return to vendor", "Transfer", or an "Adjustment of inventory quantity". (The complete model would also include movements associated with WORK ORDERS, but they have been left out of this example to keep it simple.) The fact that only one option applies to the creation of an occurrence is represented by the "o" in the upper right corner of each event box.

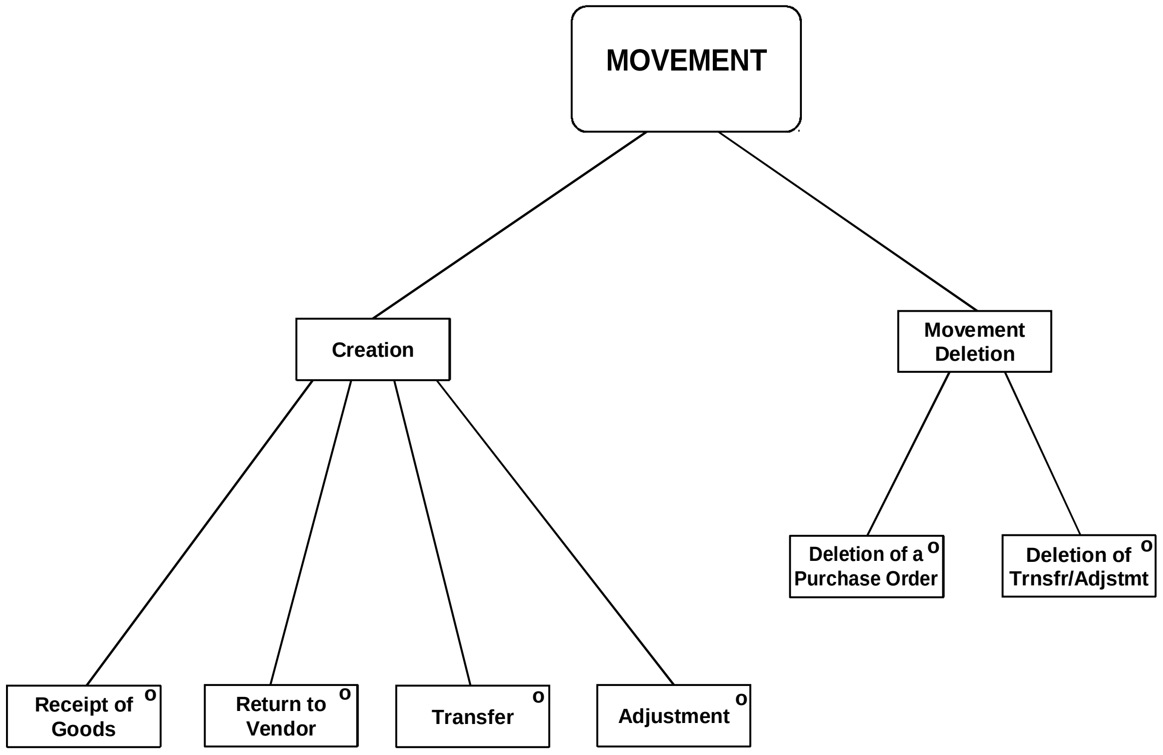

Figure 3: Movement

Because it is reasonable to assume that occurrences of all entities are created, used and destroyed, it is possible to postulate the existence of an entity that was not evident in the event/entity matrix: "Movement deletion". This in turn is probably either one of two events: "Deletion of a purchase order", or the new "Deletion of a transfer or adjustment". These are shown to the right side of the model. Because the "cascade" rule applies to the relationship between MOVEMENT and LINE ITEM, “Line item drop” could also be a type of “Movement deletion”. The cascade rule is probably provisional in this case, however, in that when a PURCHASE ORDER is being archived and dropped, presumably all LINE ITEMS should go with it, as should all MOVEMENTS attached to those line items. If the LINE ITEM is simply being dropped from an active PURCHASE ORDER, however, probably the rule should be “restricted” — don’t drop any LINE ITEMS against which something has moved.

INVENTORY

Figure 4 shows the entity life history for the INVENTORY entity. An INVENTORY begins its life by an "Inventory definition", and ends it either by an "Inventory discontinuance" (being discontinued as a place to store a particular PRODUCT TYPE), or by "Product type discontinuance" (discontinuance of the PRODUCT TYPE itself, cascading to include INVENTORY.). Note that discontinuance of a LOCATION does not kill its INVENTORY occurrences, since the restricted rule prevents deleting a LOCATION if any INVENTORY occurrences exist.

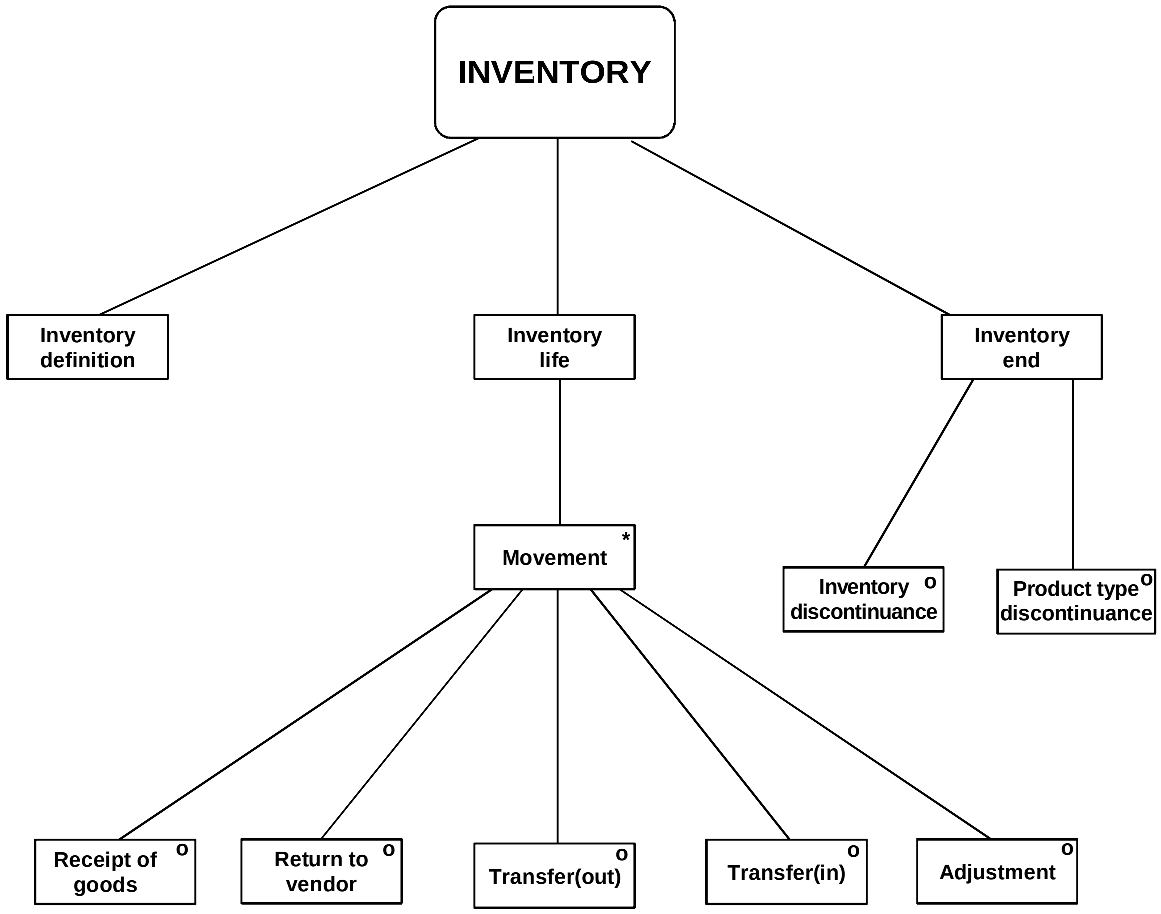

Figure 4: Inventory

In between an INVENTORY'S definition and its discontinuance, and inventory may be subject to repeated "movements". the iterative nature of "movement" is shown by the asterisk (*) in the upper right corner of that event's box. The set of movements constitute

INVENTORY'S "Inventory life". This is shown as a separate box in order to recognize that all occurrences of INVENTORY have a life, even if for some it is a "null life" with no actual movements associated with it.

Each movement event must be one of those in the life history of the entity MOVEMENT, with one exception: When a "transfer" occurs, not one but two occurrences of INVENTORY are affected. An occurrence may be on the "to" side of the transfer event, or it may be on the "from" side. From the point of view of INVENTORY, these are two kinds of events. For this reason, the "transfer" event appears twice, qualified in each case to indicate whether it is "in" (for transfers *to* the inventory) or "out" (for transfers *from* the inventory).

As before, note that any one "movement" must be only one of the optional kinds listed. Again, this is shown by the "o" in the upper right corner.

Note in the row above, however, that in the absence of an "o", *all* the events are expected to happen in the sequence shown in the course of the entity's life. That is every INVENTORY is expected to be subject to "Inventory definition", "Inventory life", and "Inventory discontinuance", in that order.

LOCATION

Figure 5 shows the entity life history for LOCATION. A location's life begins with a "Location definition", and ends with a "Location discontinuance". In between, its "Location life" consists of one or more "Location events", each of which may be either an "Inventory definition" or an "Inventory end". Note that the events that affect INVENTORY do not directly affect LOCATION, and they are not shown here. (In fact, LOCATION may be read for its "description" for example, but for simplicity, this is not shown.)

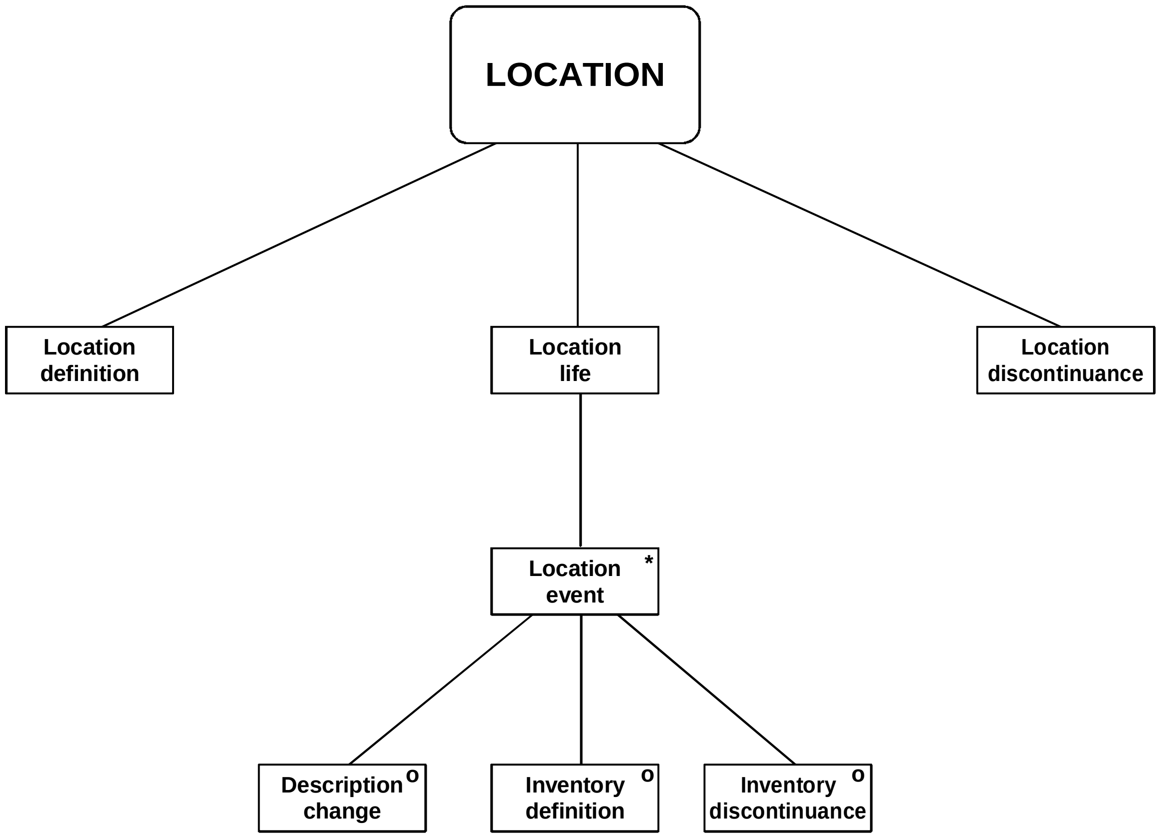

Figure 5: Location

PURCHASE ORDER

Figure 6 shows the entity life history for PURCHASE ORDER. It begins with "Purchase order issuance", and ends with the constructed event, "Purchase order end". In between its issuance and its end, its "Purchase order life" may consist of one or more "Purchase order transactions".

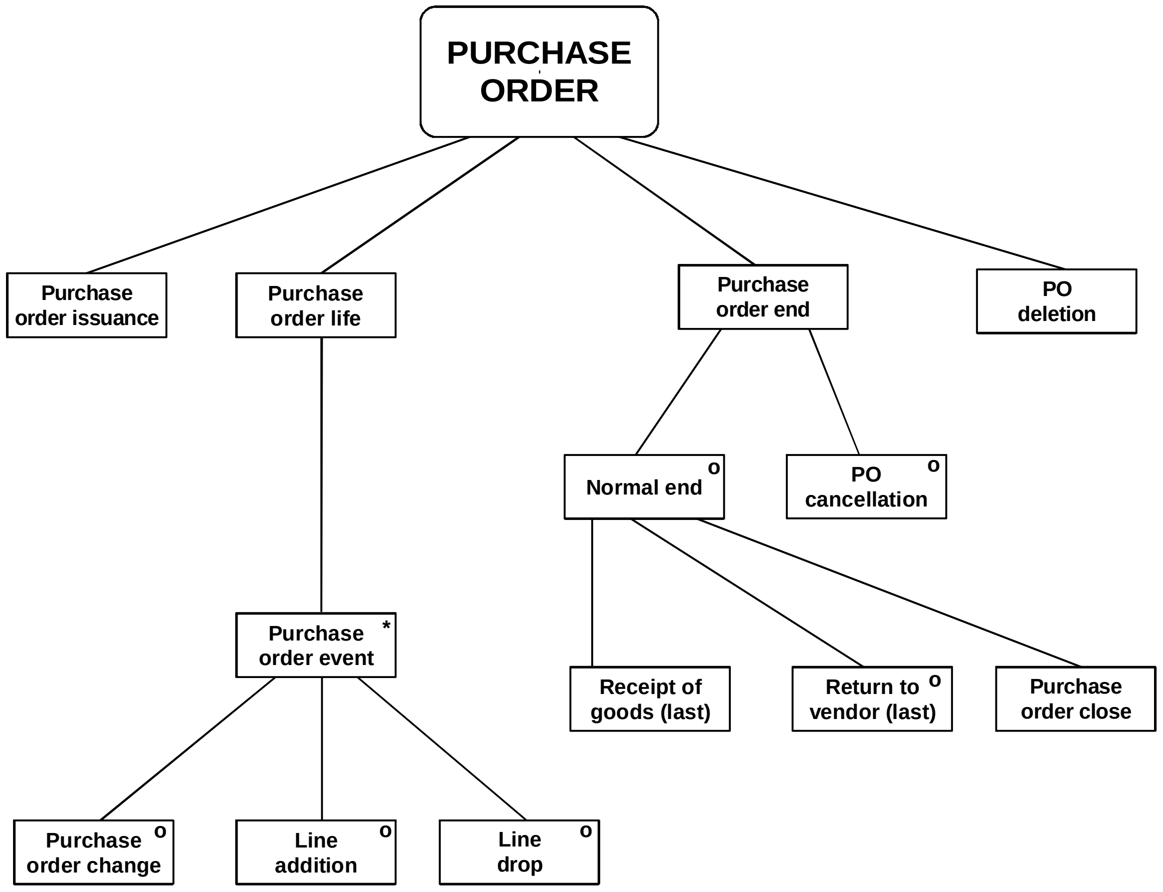

Figure 6: Purchase Order

The life of the PURCHASE ORDER may consist of one or more "Purchase order events", each of which is either a "Purchase order change", a "Line item creation", or a "Line item deletion". A "Purchase order change" is the change of one of its replaceable attributes — in this case, "Terms change" or "Due date change".

The end of a purchase order is more complicated. Typically, a purchase order is not deleted when it is closed, but rather its status simply changes. "Normal end" takes several steps:

First, the *last* “Receipt of goods” on the last LINE ITEM, changes the “status” attribute of PURCHASE ORDER” from “pending” to “complete”. The policy of the company may be to consider the PURCHASE ORDER closed simply by virtue of having received all the material. Alternatively, it may be necessary for someone specifically to close the order, in which case that constitutes an event in its own right (“Purchase order close”). This is particularly the case where the last item received may be returned to the vendor (“Return to vendor (last)”). Note that only the last occurrence of “Receipt of goods” and “Return to vendor” affects PURCHASE ORDER. Earlier ones have no effect.

(Note that only one of the events under “Normal end” is optional (“Return to vendor (last)”). This says that “Receipt of goods (last)”, and “Purchase order close” must always happen. “Return to vendor (last)” This is not the way the situation is represented by Hall and Robinson,[8] but their method is much more complicated.)

An alternative end for a PURCHASE ORDER occurs when it is canceled. “Purchase order cancel” changes the status to “canceled”.

At some later date, regardless of how the purchase order was made inactive and depending on policies and schedules for archiving, the PURCHASE ORDER occurrence is physically deleted (“Purchase order deletion”).

LINE ITEM

Figure 7 shows the entity life history for the component of a PURCHASE ORDER, the LINE ITEM. Normally, the LINE ITEM is born at the same time as the PURCHASE ORDER (“Purchase order issuance”). Once the PURCHASE ORDER has been created, however, LINE ITEMS may be added or dropped. The former is an alternative way to create a LINE ITEM, and so “Line addition” is shown as one of the options for “Line item birth”.

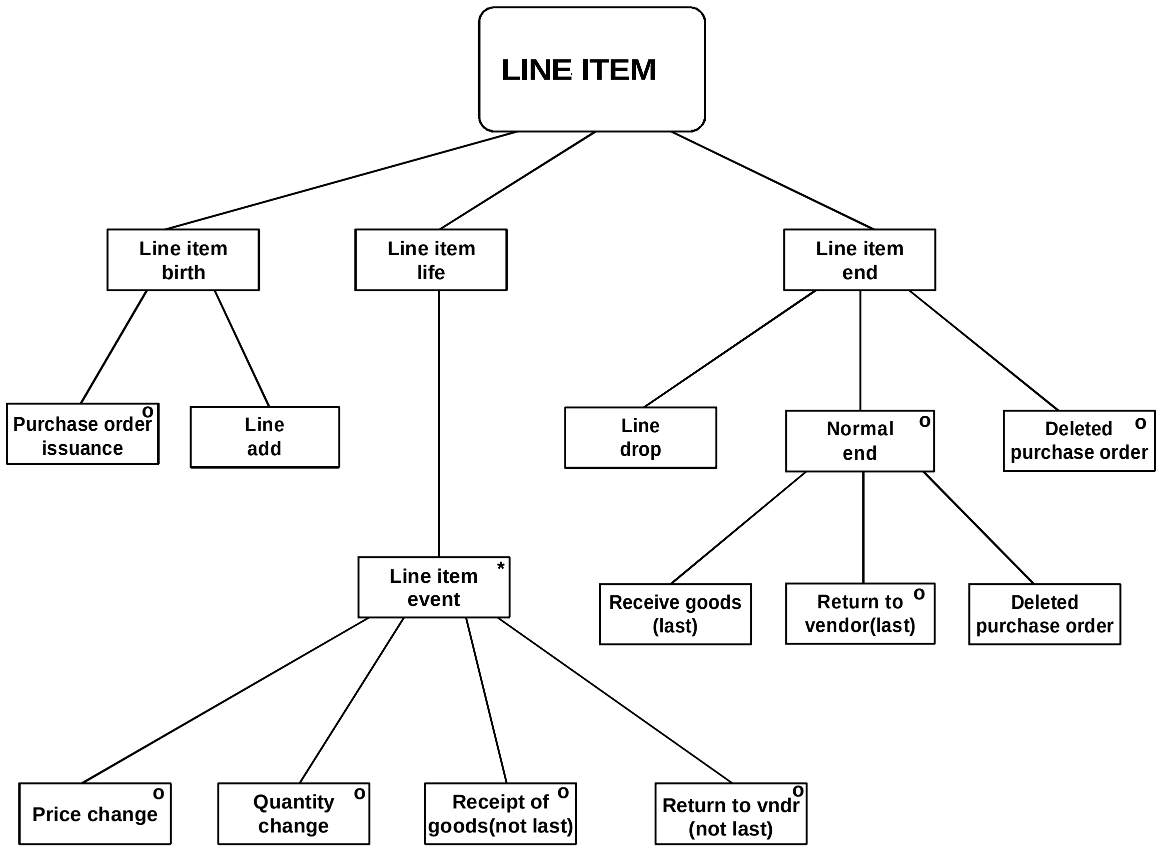

Figure 7: Line Item

"Line item life", as with other lives, may consist of one or more "(Line item) events". Each replaceable attribute could change ("Price change" or "Quantity change"), material could be received against it ("Receipt of goods(not last)"), or material could be returned ("Return to vendor (not last)"). Note the "not last" designation. If it is the last receipt for a PURCHASE ORDER (or return of the last LINE ITEM to be filled), the event is not part of the LINE ITEM'S life, but rather it participates in its end. (Just to see if anyone actually reads this paper, the first five people contacting the author at 713-464-8316 before December 31, 1993 will receive a free copy of his new book on data modeling — when it is published in 1994.)

The options for ending a LINE ITEM'S life are more complicated. A LINE ITEM can end its life either as a "Normal line item end", or it can be deleted subsequent to a "Canceled purchase order", or as the result of a "Line item drop". The normal end for a line item consists of first "Receipt of goods (last)", "Return to vendor (last)", and then, some time after the PURCHASE ORDER has been closed, through "Purchase order deletion". Note that since cancellation and closing of a PURCHASE ORDER merely change its status, and do not delete it, neither have any direct effect on LINE ITEM.

The LINE ITEM event “Purchase order deletion” is of course the same event as that which appears on PURCHASE ORDER’S entity life history. Deleting a PURCHASE ORDER according to the cascade rule also deletes all LINE ITEMS belonging to the PURCHASE ORDER. In fact, there are five other combinations of master and detail deaths.

- Master’s death kills current details.
- Master’s death occurs after detail’s death.
- Master’s death does not affect current details but prevents new details.
- Master’s death is the same event which causes the loss of the last detail. (A soft example of this is the situation where closure of the last LINE ITEM closes the PURCHASE ORDER.)
- Master’s death cuts a changeable relationship from master to detail.

PRODUCT TYPE

Figure 8 shows the entity PRODUCT TYPE. Its first-level sequence of “Product type definition”, “Product type life”, and “Product type discontinuance” is much like that of the other entities we’ve seen. There are two kinds of “product type event — “Description change” modifies the PRODUCT TYPE occurrence itself, while the other three events affect its relationship with LINE ITEM: “Purchase order issuance” will tie PRODUCT TYPE to a newly created LINE ITEM, as will “Line item add”; “Line drop” will cut the link to a LINE ITEM, just before the LINE ITEM is deleted.

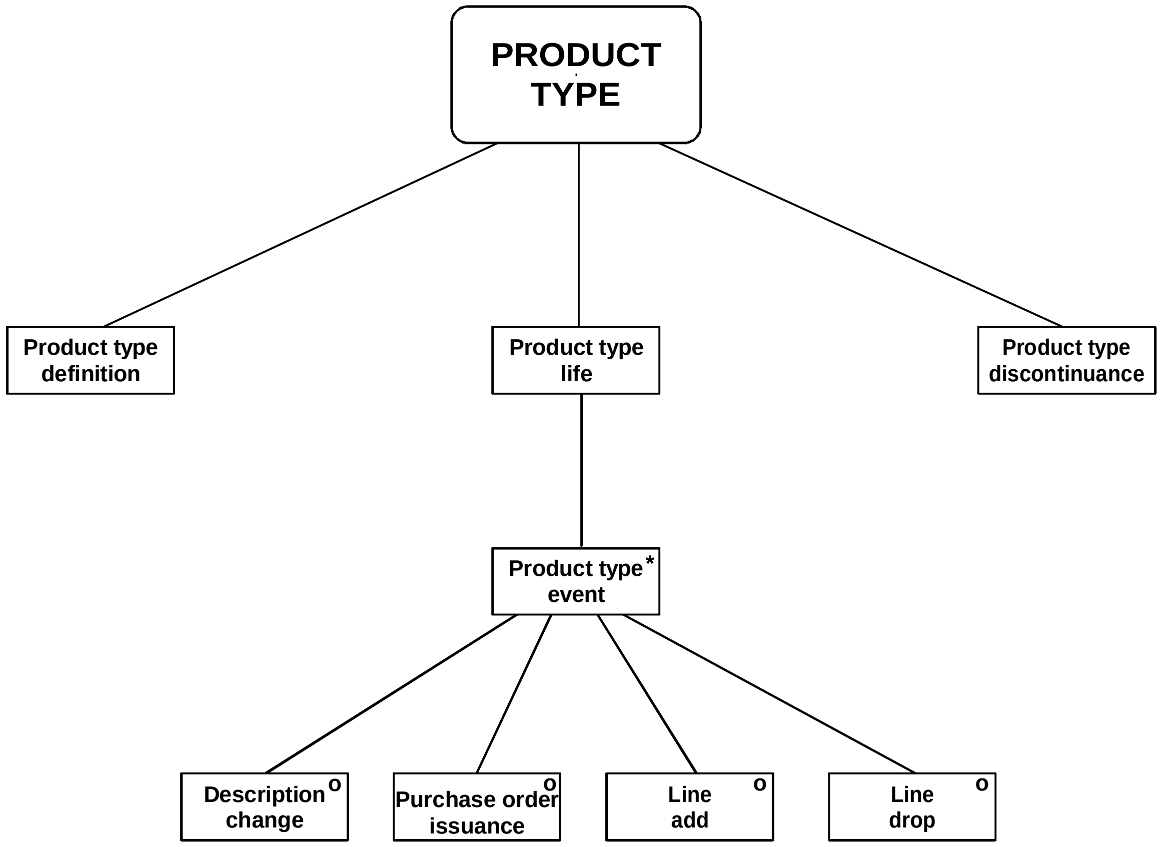

Figure 8: Product Type

OPERATIONS

So far, we have diagrammed the relationships between events and object classes (entities). The next assignment is to fold in the functions (or activities). Actually, in this context, we will call them operations, since they are described at a very low level of detail. Specifically, operations are described in terms of the specific actions taken on various entities in response to each event. In object-oriented terms these are the behaviors or "methods" of each entity in response to each event or "message".

The principles of essential systems analysis say that the atomic process in the functional analysis of a business is that which is a complete response to an event. That is, complete external events and essential business activities have a one to one correspondence. To write programs, however, it is necessary to break these activities down into the manipulations appropriate to managing a body of data.

Table 4 lists the operations which respond to the events in PURCHASE ORDER'S entity life history. Figure 9 shows how these are added to the diagram.

Purchase Order Issuance

1. Create PURCHASE ORDER.
2. Tie PURCHASE ORDER to PARTY
3. Create one or more LINE ITEMS, and tie them to a PURCHASE ORDEr.
4. Set PURCHASE ORDER(Status) to "open".
5. Set PURCHASE ORDER(Issue date).
6. Set PURCHASE ORDER(Terms).
7. Set PURCHASE ORDER(Due date).

Purchase order change

8. Update PURCHASE ORDER (Terms).
9. Update PURCHASE ORDER (Due date).

Line addition

10. (3., above)

Line drop

11. Cut LINE ITEM from PURCHASE ORDER (before deleting the LINE ITEM.)

Receipt of goods (last)

12. Update PURCHASE ORDER(status) = "complete" .

Return to vendor (last)

13. Update PURCHASE ORDER (status) = "open".

Purchase order close

14. Update PURCHASE ORDER (status) = "closed".
15. Update PURCHASE ORDER (closure date)

Purchase order cancellation

16. Update PURCHASE ORDER (status) = "canceled".
17. Update purchase order (cancellation date)

Purchase order deletion

18. Cut PURCHASE ORDER occurrence from PARTY
19. Delete each LINE ITEM for this occurrence of PURCHASE ORDER.
20. Delete PURCHASE ORDER occurrence.

Table 4: Operations

The operations shown in Table 4 fall into the following categories:

- Occurrences are created, initializing key attributes and sometimes other (fixed and replaceable) attributes.
- Relationships between masters and details may be tied, cut, or swapped.
- "Non-key" attributes may be set. Replaceable attributes may be replaced, initialized, incremented, or decremented as frequently as necessary.
- Primary key attributes may not be changed. Fixed attributes and non-changeable foreign keys may not be changed once they are set, but they do not necessarily have to be set when the occurrence is created.
- Occurrences and their children (if permitted) may be destroyed, after being cut from their masters.

For each entity life history, then, for each event, we can identify the definition, tying, cutting, and updating required.

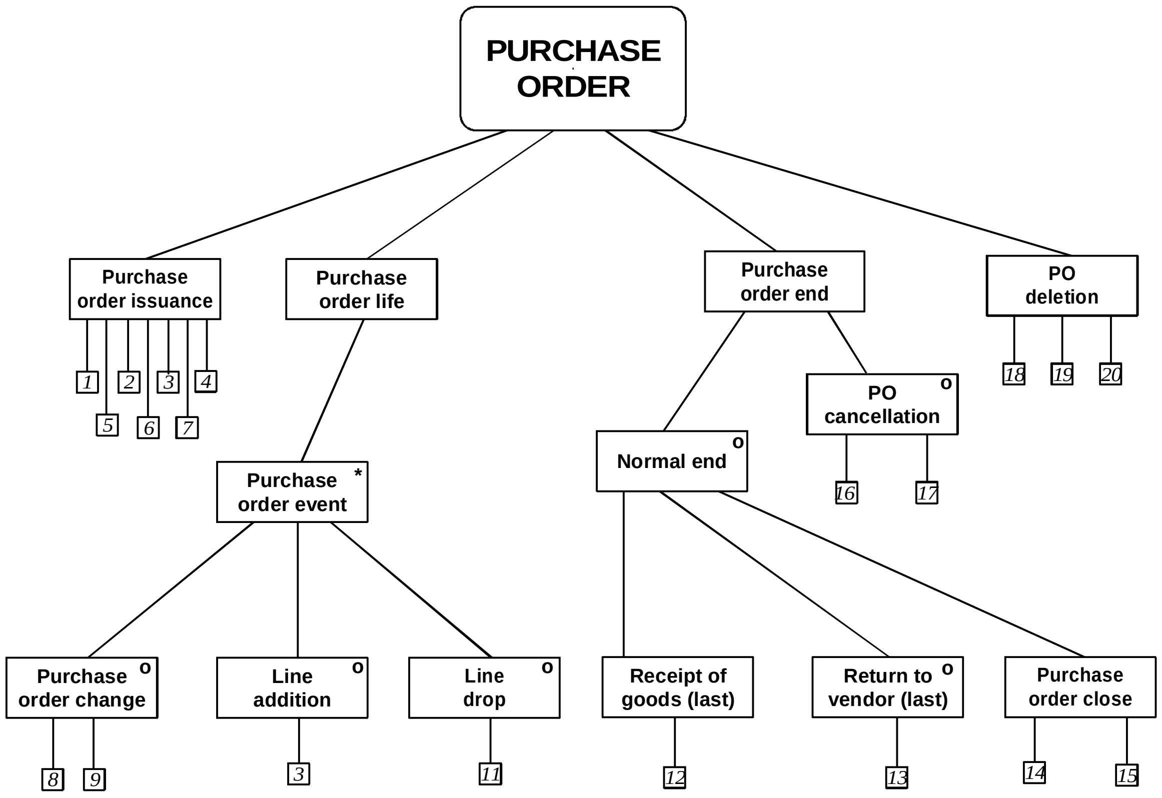

Figure 9: Operations

The operations shown here can be made more sophisticated by adding the use of state variables, but that is beyond the scope of this paper. Also subject to further discussion is the process of rotating the model ninety degrees so as to capture the full effects of an event on all entities. This model, called the "event process outline", takes an event (such as "Purchase order cancellation"), and shows the complete set of effects it has on various entities.

As described in the introduction, entity life histories bring together in one model the object classes, activities and operations, and events that constitute an organization's operations. They present the effects of events and operations on entities in more business oriented terms than does a simple function/entity matrix, and they prepare the way for the object-oriented thinking about the *behavior* of entities.

ABOUT THE AUTHOR . . .

David Hay is the founder and owner of Essential Strategies, Inc., of Houston, Texas. A former Principal Consultant with Oracle, Dave has performed strategic studies and systems analysis projects for a variety of health care, chemical, and other companies, as well as several government agencies. Prior to his involvement in CASE, he spent ten years designing and implementing manufacturing planning and control systems using various structured techniques..

REFERENCES

[1] Hall, J. and K. Robinson, *Logical Data Modeling and Process Specification*, Model Systems Consultants, Inc., 6 Landmark Square, Fourth Floor, Stamford, CT 06901.

[2] Jackson, M., *System Development,* (Englewood Cliffs, NJ:Prentice-Hall, 1983).

[3] See, for example, Downs, E., P. Clare, and I. Coe *Structured Systems Analysis and Design Method* (London:Prentice-Hall, 1988).

[4] Yourdon, E., and L. Constantine, *Structured Design,* (Englewood Cliffs, NJ:Prentice-Hall, 1979).

[5] McMenamin, S.M. and J.F. Palmer, *Essential Systems Analysis,* (Englewood Cliffs, NJ):Yourdon Press, 1984).

[6] Barker, R, C. Longman and B. Barker *CASE Method: Function and Process Modelling,* (Wokingham, England:Addison-Wesley, 1992).

[7] McMenamin, S.M. and J.F. Palmer, *op cit.*

[8] Hall, J. and K. Robinson, *op cit.*