BUSINESS EVENT METHODOLOGY

BY BRIAN DICKINSON
LOGICAL CONCLUSIONS, INC

EXCERPTED FROM *Creating Customer Focused Organizations.*[1].

The world thus appears as a complicated tissue of events, in which connections of different kinds alternate or overlap and thereby determine the texture of the whole.
Werner Heisenberg
Physics and Philosophy

We will examine the central concept of Organizational Events in this chapter. We will see that functional partitioning based on some of these Organizational Events provides an objective basis for the structure of an organization.

There are five types of Organizational Events:

- Strategic Events,
- System Events,
- Business Events,
- Regulatory Events, and
- Dependent Events.

However, the only ones that go on the Business Model are Business Events, Regulatory Events, and Dependent Events.

From here on in, we'll call these three the Business Model Events. These three types of Events occur at the boundary of our organization. This is our Event Horizon (see Figure 1).

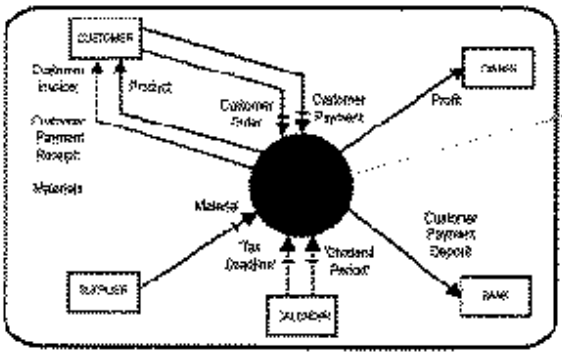


Figure 1 – The Boundary of the Organization

We must recognize the Strategic and System Events so they don't cloud our Business Model. These two Event types are somewhat internal; they come from our organization's Strategic Planners and Designers.

We must always keep in mind when determining Event types that Business Model Events are context driven. In other words, aspects that are business issues to one organization may be system issues to another organization. Let me clarify this after we've defined the five types of Events.

I want you to keep in mind the fundamental characteristics of all systems as stated in The Nature of Systems Chapter. We will use the Stimulus-Response and Process-Memory structure from that chapter as an underlying tenet throughout this chapter.

THE FOUNDATION OF THE EVENT METHODOLOGY

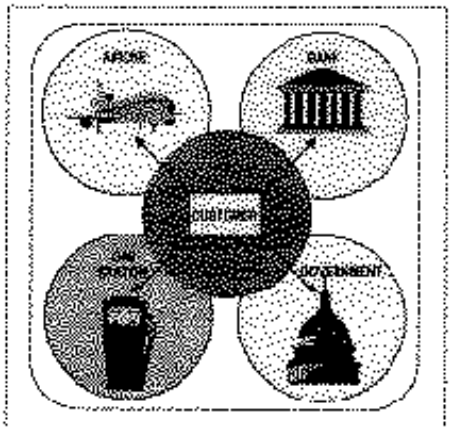
There's a multitude of demands people make on the world of business and each requires a different response to satisfy it. On the other hand, any one organization responds to only a small subset of those demands. (As an added benefit, in vertical, non-diversified organizations, we can usually obtain mass reusability in our processing and data across these demands.)

From one point of view we could say that any organization has simply decided to respond to a set of customer needs (Events) that occur in the external world. As examples:

- An airline company decides to respond to its customers' travel needs in which they, and/or goods, must be moved.
- A bank decides to respond to its customers' needs for storing and manipulating money.

- A fire department responds to a fire in the environment (the department's customer).
- Logical Conclusions, Inc. has decided to respond to our customers' needs for consulting or training in subjects such as Customer Focused Engineering.

Figure 2 – All Organizations Respond to a Set of Customer Needs



We never know within our organization why the demand was generated. That's why we classify it as outside of our organization.

- Why a customer wants to withdraw funds from our bank for example is literally none of our business; a funds withdrawl could be made in order to go on vacation, pay a bill, or just have expense money for the day.

A system (a manual operation, a computer system, or even our entire implemented organization) has no control over external customers' needs. This does not mean we should not "think out of the box" and ask our customers if we can assist them beyond our current boundary. The customer has the ultimate control in initiating a stimulus to our organization. At the same time, however, our organization is obliged to respond to stimuli caused by our customer's needs.

Without stimulus from the outside, our organization and its systems remain inactive and are essentially meaningless, and our organization will eventually close down. No organization is self-perpetuating. Even non-profit government organizations respond to external needs as their reason for existing.

All organizations rely on their customer's Events as the reasons for existing.

A LITTLE HISTORY OF THE CONCEPT OF EVENT PARTITIONING

In the 1970s the Systems Engineering techniques that were being proffered were somewhat disjoint, mainly because the ideas originated with different authors (and their sales staffs). During that same decade, I worked at a young company involved in generating and spreading these new Systems Engineering ideas. In those earlier years of teaching new engineering tools we didn't quite have the techniques down. For example, we knew how to draw a Data Flow Diagram and we understood the usage rules of each symbol, but we didn't give much advice on how to produce a well-partitioned Business Model.

The size of the drafting surface—a page—and the human limitation of keeping track of seven plus or minus two things per page were somewhat the extent of our partitioning rules.

In my seminars I could convince my data processing students that there was a better way to partition systems because almost all existing partitioning of programs and systems was mainly at the discretion of some D.P. professionals. (D.P. was a new industry and almost all programmers were allowed to partition their programs and systems as they saw fit.) This arbitrary partitioning was not so true for the business community. Of the seminars I taught in the 1970s, the ones in which I taught analysis were the only ones where I would expect a business person to be present. They at least had a longer history and established reasons for how and why they partitioned their departments, people, and tasks. I had to have good reasons for repartitioning these established organizational boundaries.

- I and a few of my colleagues who taught analysis seminars at that time used a technique called Stimulus-Response Modeling as a means of partitioning a business Data Flow Diagram. In 1984 two of my colleagues at Yourdon, Inc., Steve McMenamin and John Palmer, used the term "Event Partitioning" in their book, Essential Systems Analysis, for a similar concept. This was a means of getting away from any of the physical/design partitioning of the old environment. We used a "string-of-pearls" analogy to illustrate a stimulus data flow triggering a chain of continuous processes connected by intermediate data (indicated by lines and circles on a Data Flow Diagram).

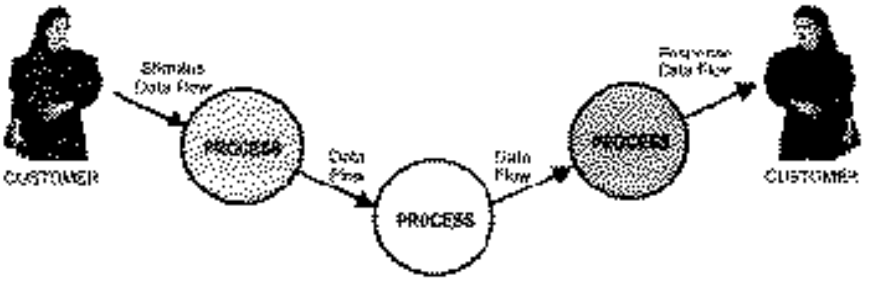


Figure 3 – The String of Pearls

In Figure 3 we see an external stimulus initiating the execution of the set of processes (the string of pearls). The intermediate data and its processing were required to respond to the Event that produced the stimulus. Once initiated, the set of processes runs its course without further stimulus until a response is returned to the outside world. Because all these processes, and only these processes, are triggered, we can see that they make up a highly functional partition in that they perform a "single-minded" overall function from the point of view of the external Customer.

This basic concept has evolved (over 20 years) into the contents of this book (*Creating Customer Focused Organizations*) built around the central theme of the Business Event.

THE BUSINESS EVENT METHODOLOGY AND ITS PLAYERS

Before going any further, let me establish a Common Platform of Understanding regarding the term Business Event Methodology. I have heard many discussions and/or meetings go awry when people use the terms "methods" and "methodology" interchangeably and misuse the word "methodology." Webster's Dictionary defines "methodology" as: *A set or system of methods, principles, and rules for regulating a given discipline.*

So a methodology recommends a cohesive set of methods which in turn usually recommend a set of models (See Figure 4).

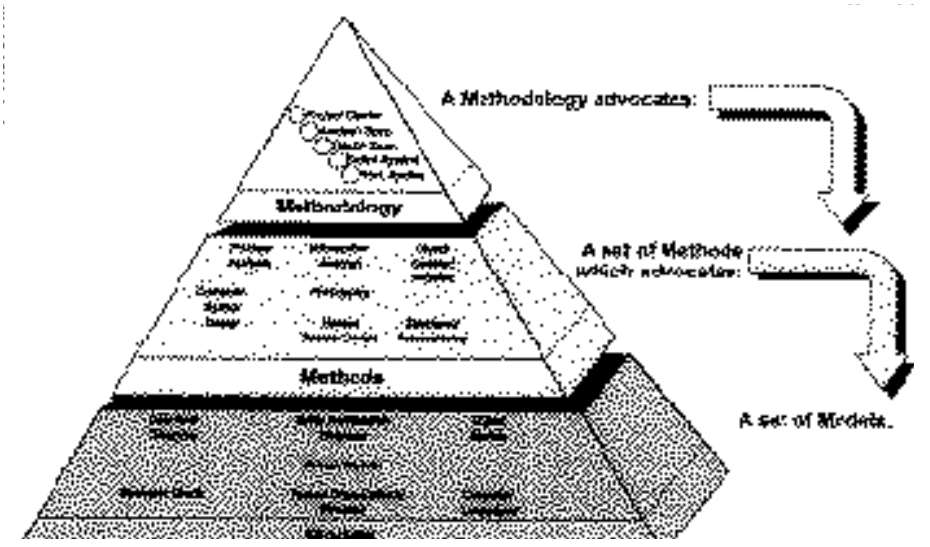


Figure 4 – An Example of Methodology, Methods, & Models

At a high level the methods look the same for any project: conduct analysis, design, and implementation, etc. How you perform these methods is different in the Business Event

Methodology. The individual methods utilized within the high-level phases are typically selected by each project. Just make sure when creating a Customer Focused Organization that the methods are engineering methods and the intermediate deliverable (the model) from one method flows naturally into the next method.

There are a number of "players" (see Table 1) and tasks associated with those players in the Business Event Methodology. Remember that these players could be different people in a development project, or they could be one person wearing different "hats."

- The customer has the need.
- The Strategic Planner identifies if the organization will respond to the need. The Business Policy Creator interprets the need into the specific set of data and processing the organization will use to respond to the need. (In other words, the organization's decision-makers determine the business policy.)
- The Systems Analyst models/documents this business policy.
- The Systems Designer invents the design to implement this business policy.
- The Systems Builders (Technical Writers and Programmers) implement this business policy.
- And finally, the Production Systems (People, Technology, etc.) accomplish the customers' need through time.

Customer	Strategic Planner/ Business Policy Creator	Systems Analyst	Systems Designer	Systems Builder	Production Systems (People & Technology)
→ Has the Business Event Need	→ Creates the Business Policy to Satisfy the Customer's Need	→ Analyzes and Models the Business Policy	→ Designs the Implementation of the Business Policy	→ Implements the Business Policy	Accomplishes the Customer's Need through Time

Table 1 – The Players in the Business Event Methodology

THE TYPES OF ORGANIZATIONAL EVENTS

OK. You've reached the most important part of the chapter. I had to state all of that other stuff in the preceding paragraphs to set the foundation for this part. In the Business Event Methodology we identify five types of Organizational Events:

- Strategic Events
- System Events
- Business Events
- Regulatory Events
- Dependent Events

Determining the type of Event we are analyzing is important because we want to concentrate on the ones that make up our true business, i.e., form our Business Model. Some Organizational Events don't belong on the model at all, while another type actually changes the model. Before defining these types of Organizational Events, I want to make sure you don't confuse an Event with its stimulus and implementation. A phone call, a piece of mail, a fax, or a customer walking up to a counter are all implementation issues. Phones, the mail, a fax, or a personal visit are just the designed means for communicating the customer's need (the Event).

Identifying Organizational Events via where they Occur

Where the Event occurs (i.e., originates) often dictates the type of Event:

- Strategic Events typically originate at our organization's high-level management (although the management may have initiated the Strategic Event based on our competition doing something to which we need to respond). The management may also be responding to an outside agency whose laws affect our organization—more on this later. Strategic Events can't be ignored, but they are not the main focus of the Business Event Methodology.
- System Events originate within the organization. They are invented and imposed on the area of business to meet management and design needs. They will always be associated with an implemented system.
- Business Events originate at our true external customers. Business Events are the most important Event to the Customer Focused Engineer.
- Regulatory Events originate at a government or external regulatory agency. They are imposed on the area of business to meet legal operating requirements.
- Dependent Events typically originate at the vendors used by our organization. Dependent Events are the result of "farming out" part of the business in the past.

Organizational Events also have a hierarchy in terms of their of importance to the Business Event Methodology (see Figure 5).

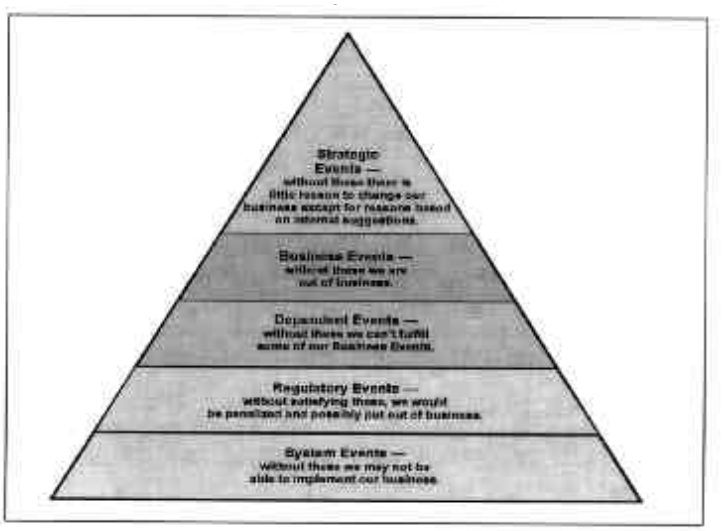


Figure 5 – The Hierarchical Importance of Events

Every organization I have studied responds to all these Event types. You may have already guessed that when we create a Customer Focused Organization, we want to focus on the Business Events. They will form the bedrock of our Business Model. However, we have some Event types that get in the way of forming our Business Model and others that we must also include in our Business Model (along with Business Events). Figure 6 shows the Customer Focused Engineer's Event filtering and categorizing task. Notice that this diagram is indicating that we're trying to focus on true Business Events as our organization's mission, but we have Strategic Events and System Events in the way of our business view. We should filter these out first. We will keep Dependent and Regulatory Events on our Business Model but we should identify and categorize them on our model as such. Even though they appear on the Business Model, they are less important than Business Events because they are not our main focus for "keeping the doors open."

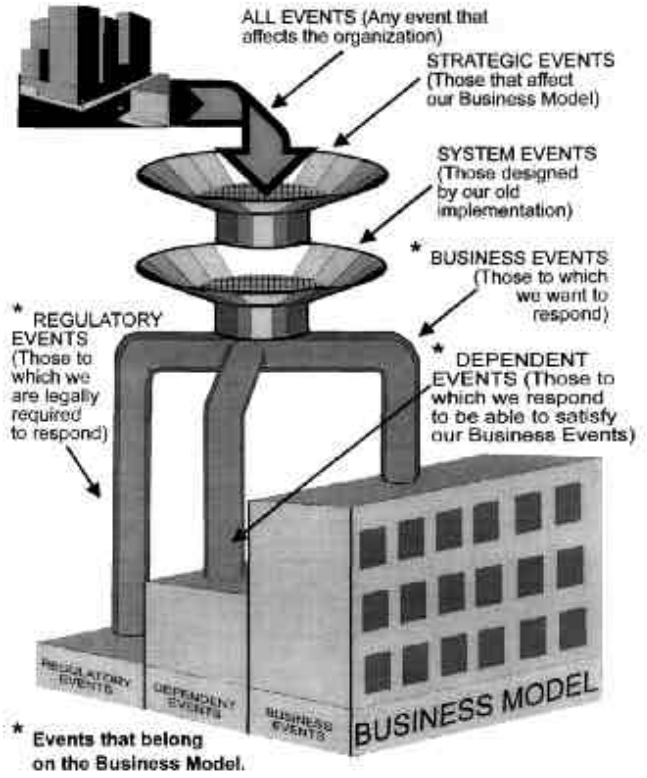


Figure 6 – Filtering and Categorizing Events

You could create separate models for Dependent Events and Regulatory Events, but I recommend you don't when you're new to this methodology because there will be shared processing and memory between these three types of Organizational Events. Although, it may be a good idea to indicate the different types of Organizational Events that do belong on the Business Model (Business, Dependent, and Regulatory) by using different colors or codes.

I have found it helpful to use the questions and answers in Figure 7 as a quick reference for determining the type of Event or aspect being analyzed.

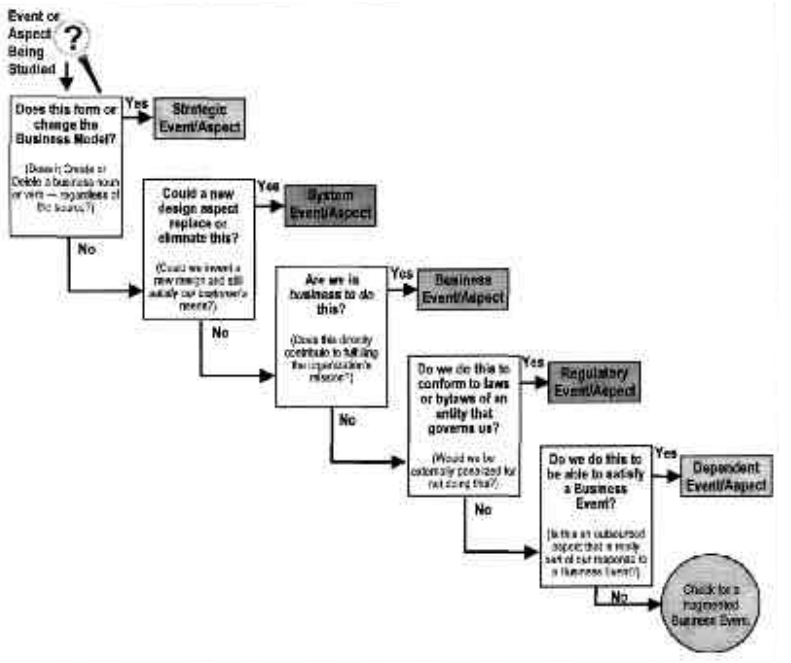


Figure 7 – Determining Event/Aspect Types

UNFRAGMENTING EVENTS FROM OLD DESIGN TRAPS

Of the five types of Events identified in this chapter, Business Events are the most important in our effort to create a Customer Focused Organization. It's relatively easy to spot our organization's Business Events since they always originate at our true external customers. However, it's usually not so easy to follow the Stimulus from an Event through its Processing and Memory in a typical organization. The larger and older an organization, the more the trail will resemble a long, winding calf path.

There are two stages to unfragmenting Events from old design traps:

- Dis-covering the real business processing and data by removing the cover placed on these by the old design, and
- Unfragmenting our response to a Business Event into a cohesive partition to satisfy the customer's need.

The fragmentation of Business Events will typically occur across many automated and manual boundaries. If we look upon department and computer system partitioning as vertical slices in an organization (see Figure 8), then Business Event Partitioning will probably take the opposite view, that of horizontal partitioning.

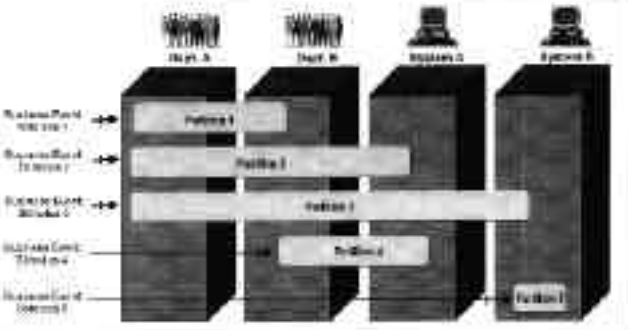


Figure 8 – Traditional vs. Business Event Partitioning

Notice in Figure 8 Business Event #5 is completely subsumed within an existing designed system boundary. Hopefully, we will find some of these, but they will usually be coincidental.

- For example, the Regulatory Event for "End-of-Year Tax Reporting" is completely within the Accounts Departnebt because they happen to have all of the financial data and this Event only requires financial data and processing.

Because a Business Event will probably have been broken up by the traditional design of an organization, we will find a number of "traps" that will make it difficult for us in "discovering" our true (and whole) Business Event Partitions. Let me identify some of these traps.

Beware of Fragmented Events (Design Trap #1)

As a Customer Focused Engineer we must be able to recognize "fragmented" Business Events that occur at internal design boundaries. These are often places where a Business Event stopped in relation to a past area of study, and hence, a point where a Business Event got fragmented. This Internal Interface may be an internal person (such as a clerk in Accounting), or it could be a calendar that stimulates a batch of data to be input into a portion of the business.

In the past, many designers believed they had to break a Business Event into many individual transactions based on the restrictions of batch processing, central mainframe capacity, or, from a manual point of view, on human skills and memory.

When performing analysis and identifying an interface where a Business Event occurs, we must first ask whether the person or system stimulating us is external to our organization and not just to the portion under study. In other words, are they truly a customer—someone to whom we need to respond in order to satisfy our strategic mission? (Remember, this customer could be a place on Earth where lightening starts a fire, or even the calendar reaching a significant point in time to which our business must respond.) If the origin of the Event is internal and it is there only because of project scope or for design reasons, we must be careful not to dismiss this fragmented Business Event as a System Event because it looks as though

we are in control of it. We must also make sure we bring back together fragmented Business Events when we create our Business Model.

It's when we study a subset of our organization that we start to confuse System Events with fragmented Business Events. This also can happen when we break up our project into different project teams to study subsets of the organization.

If you start from the organization's true outside boundary with a Business Model Event List, then any Business Event should be modeled in its entirety without any fragmentation. Therefore, we should not confuse System Events and fragmented Business Events in an organization-wide Customer Focused Engineering project context. Figure 9 shows a complete Business Event that was fragmented.

This may make us look upon the processing triggered by the System Event Stimulus "End-of-Day" as unnecessary System Event processing; however, it is really part of the fragmented Business Event triggered by the customer.

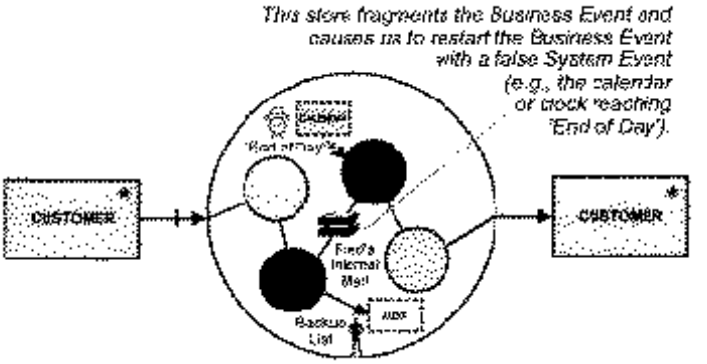


Figure 9 – A Fragmented Business Event Causes a System Event

- The classic "real-world" example of this is a stock control system that is triggered by the "Calendar" (appearing to be a System Event); however, ordering "Materials" (in an organization that sells "Materials") is not an aspect of design. This is where reording got fragmented from a real Business such as: "Customer Wants to Purchase Our Materials" resulting in "Materials" being removed from the "Warehouse." This is the point at which "Materials" could fall below the necessary reorder levels. Therefore, it is the most natural time and place to reorder "Materials."

Transaction Types and Fragmented Events

Transaction type fields and codes alert us to look for fragmented Business Events.

- For example, in one of my seminars I had a Data Processing student ask, "Isn't a Business Event the same as what we call a transaction in batch systems?" For the Customer Focused Engineer, it's important to keep the notion of Business Events clear of transaction issues that pertain to design and implementation. A Business Event is not a transaction. Transactions belong to the world of the designer or programmer.

Typically, "transaction thinking" is a single Business Event broken down and lumped together with other kinds of Business Events to be processed in transaction batches. A dead give-away that a Business Event has been broken apart and/or batched is when we see "transaction type" or "request number", etc. fields. A transaction-type field is used to keep track of each Business Event within a batch and its associated processing (e.g., a person or program has to interrogate the transaction type field before they or it knows what type of processing needs to be executed).

We call these broken apart events "Fragmented Business Events." The important task for the Customer Focused Engineer is to bring together the fragmented parts of a Business Event. So, some "system" stimuli will be in place to support fragmented Business Events. These stimuli typically will be associated with a transaction type field or a colored form.

- For example, we may have a batch process triggered by the start of the business day and this process may start at that time because that's when people show up to work in the Accounts Department. This process really should have been part of our Response to a Customer Order where the flow triggers the processing in both the Order Department and the Accounts Department and ignores the department boundaries entirely.

This transaction view is one of the hardest habits to break. People who have this batch-oriented view of manual or automated systems want to put a distribution process at the beginning of their Business Process Model (a system processing issue rather than a business issue). This process manifests itself in the manual system world as the Customer Service/ Help Desk and in a computer system as a Transaction Center module at the beginning of the program. In non-Customer Focused systems this distribution process is needed to separate out the lumped together Business Event Stimuli (see Figure 10).

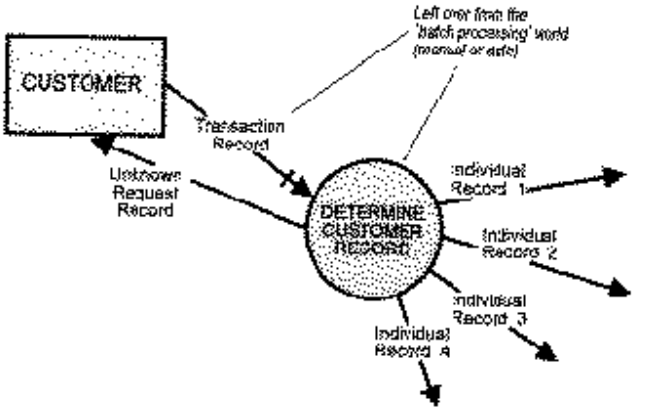


Figure 10 – Erroneous Process Model "Transaction Warp"

I always ask these "transaction warped" students to try to write the business logic inside that distribution process. This is invariably a fruitless task because the logic needs a transaction type indicator that was invented by the designer, rather than requested by the business or the customer, and all of the logic is system/design oriented.

There is a need to differentiate between Internal and External Interfaces on a model. If we're using a Data Flow Diagram to model our business, then an External Interface is a solid-lined rectangle; if the interface is Internal, I recommend a dashed-lined rectangle. I introduced this into the Data Flow Diagram symbology in Chapter 5. If you use another model, then I believe it would be helpful to extend that model's symbology to address this need.

- For example, on an Object Oriented Model you may have Messages that are internally stimulated represented differently than externally stimulated Messages.

My reason for putting forth the idea of an Internal Interface and for recommending the introduction of new symbols is to point out where future projects need to link to other parts of a fragmented Business Event caused by the data flow terminating at an Internal Interface.

At some future point we would eliminate these Internal Interfaces and implement complete Business Events across the whole organization; this should be part of the Strategic Plan of our organization. The timing of this future point depends on the implementation of a Strategic Plan based on Business Events. After the Strategic Plan is completely implemented, there will be no Internal Interfaces on our organization's Business Model. Also, under this unifying discipline we would not continue our old, historical error of implementing fragmented Business Events. Note that if we do Pre-Engineering (as discussed in the first chapter), we wouldn't introduce these fragmented Business Event driven Internal Interfaces in the first place.

Beware of Bundled Events (Design Trap #2)

Just as "Edit-Update-Print" is a poor basis for partitioning because it fragments business policy, so is collecting all similar processes together, such as Create, Retrieve, Update, and Delete to form monster partitions.

- For example, we might collect all Create Events (such as "Create New Customer" or "Create New Vendor") together to form a generalized Create Program. This view, shown in Figure 11, could come from focusing only on data instead of both data and processing together. This is not a specific Business Event view, and if implemented as four design partitions (programs), we would have similar maintenance problems as with the Edit, Update, and Print partitioning.

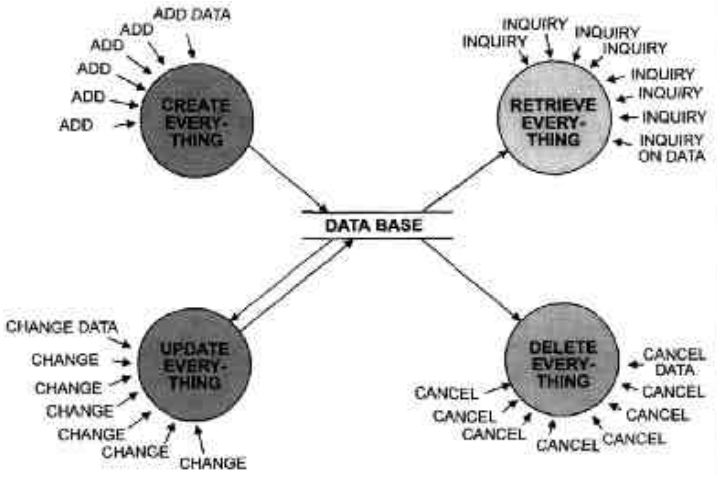


Figure 11 – Bundled Business Events

Carried to its extreme, we could incorrectly multiply the number of Data Elements/ Fields and Entities in our business by four to incorrectly derive the total number of Events in our business. (I hope that you can see that this does not represent a "business view" but rather a "data-only oriented" point of view.)

If we talk about quality as "conformance to customer requirements," to obtain the ultimate customer satisfaction, we need to satisfy the customer's specific requirements. If we are lumping together different types of orders from customers into a general Customer Order, we will have bundled together a set of separate Business Events. This leads, in design, to what we find in most businesses today—the standard order form/screen and the standard processing of that order form/screen. The processing of a generic order may be implemented using human beings or an automated system. In either case, we end up with a form or screen containing data fields that should never be filled out for a Special Order, but which definitely

need to be filled out for a Delivery Order. Also, different parts may never have to be completed for an Advance Order while other parts may have to be filled out for a Pre-paid Order. What happens is, we end up with a global form or screen and the global processing needed to satisfy that form or screen. Often the person who fills out the form or screen has to conduct a long-winded procedure to overcome the design of the form or screen.

A problem also occurs when shared logic is initiated for many different Events that were usually bundled together. Unfortunately this makes it difficult to modify procedures and/or computer code when changes are needed because there is so much procedure and code (design logic) associated with the fragmentation. This design logic gets in the way of the business logic.

The logic (in computer code and procedure manuals) is usually strewn with what I call designer "IFs." The Business logic is cluttered with "IF" statements where we try to figure out which one of the bundled Events use this logic (see Figure 12).

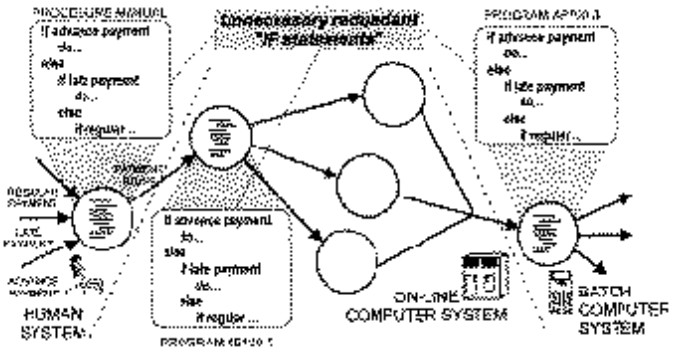


Figure 12 – Designer "Ifs" in Programs and Procedures

It gets worse, of course, when we have Fragmented and Bundled Events combined. This is what we find in the classic, historically/hysterically partitioned Edit All—Update All—Print All computer programs. We also find the equivalent historical partitioning in a manual environment in areas such as Order Entry, Accounting, and Invoicing (see Figure 13).

Figure 13 shows the evolution of Business Event modeling that would occur in the process of conducting Customer Focused Engineering on an organization that was originally historically/hysterically partitioned, causing fragmentation and the bundling of Events. This is represented in the model shown at the top of Figure 13.

The middle model shows a partially Systems Engineered organization, and finally the bottom model shows a Customer Focused Organization completely restructured with complete Event Partitions.

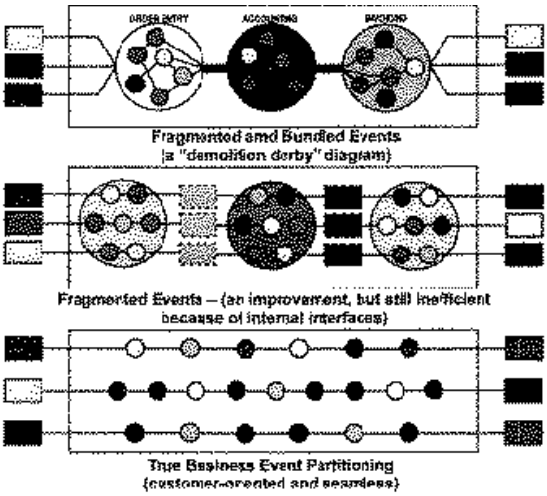


Figure 13 – Evolving to a Seamless, Customer Focused Organization

Note that the Internal Interfaces (shown as dashed rectangles on the interim model) are the result of not being able to conduct Customer Focused Engineering on the entire organization at one time. Also, the first two models in Figure 13 show where the designer complicated and slowed down the response to the Customer by bundling and/or fragmenting the Business Events.

Beware of Historical Events (Design Trap #3)

We should even guard against believing that what was a Business Event in the old system is really a Business Event that was initiated by the original customer. The last analyst (or Strategic Planner) may have dictated what a customer was allowed to do with the organization's old systems.

If the inventor of the business logic in place today had not done some type of customer survey when putting the old logic in place, then our customers may just be putting up with our limited Events until something better comes along. We don't want to create Customer Focused Systems in which our customers aren't really interested. We need to "think out of the box" on this one.

- For example, no one in the phone company asked me if I wanted to keep a thick directory of Business Information (the Yellow Pages) in my house; nor was I asked the order in which I would like the directory to be arranged. The phone company also makes me access their systems (manual or automated) with a false design key—a phone number.

Of course, there is a problem with asking a customer what they require of our organization. Many times our customer is trapped in a design world also and we may get a request to simply improve our existing designs—not new requirements.

Beware of Fragmented & Bundled Data Stores (Design Trap #4)

Just as we have old design traps for fragmented and bundled Business Events, we have the same design traps for Stored Data.

Beware of Fragmented Stores

It is typical to find fragmented "data sets" (i.e., data Entities) across old designer stores.

- For example, information on a "Customer" could be scattered across "Sales Files," "Purchased Products Files," and "Service Agreement Files" as in Figure 14. If a "Customer" sends in a change of address, all three files would need to be updated. The analyst must bring these Data Elements together into a cohesive data grouping (i.e., an Entity) when repartitioning stores.

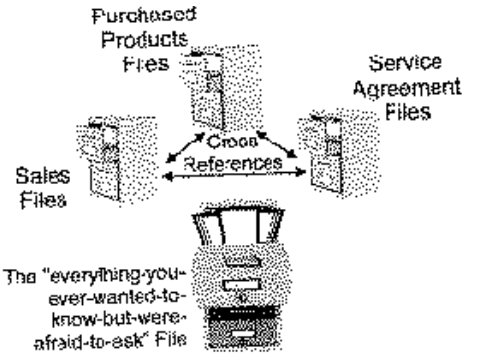


Figure 14 – An Example of Fragmented Manual Stores

Beware of Bundled Data

As we saw in the chapter "System Archaeology," bundled stores arise when the previous designer grouped separate sets of data together, possibly for the sake of efficiency. This is very common of the files used in "Edit-Update-Print" computer systems, because this data partitioning was based on process bundling for technology and efficiency. The analyst must break up these bundled stores (typified on the left side of Figure 15), and assign their Data Elements to cohesive data groupings (i.e., Entities) shown on the right side of Figure 15. We do this by observing the access key to any individual Data Element.

- For example, “Customer Delivery Address” may be retrieved via “Customer Number” in the “Purchased Products File,” “Customer Address” may retrieved via “Customer Date of Birth” in the “Sales File,” and “Customer Billing Address” via the “Customer ID” in the “Service Agreements File.”

This task typically is not as easy as in the above example because Customer Information may be stored and accessed via Customer Name and Address, Social Security Number, Customer Phone Number, or Driver’s License Number, Passport Number, etc. The task will be made easier if we already have formed a Business Event Memory store as defined in the “Partitioning by Business Events” chapter and used a specific naming standard for each Data Element in those stores.

We need to take care of this trap by unfragmenting and unbundling the data to ensure data conservation as described in the “Achieving Organizational Process and Data Integrity” chapter.

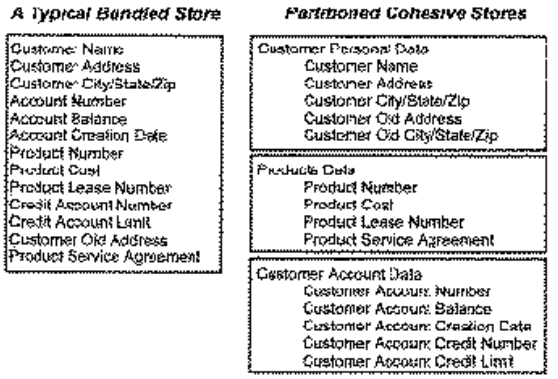


Figure 15 – Bundled vs. Cohesive Stores

What we are aiming at is a set of stores that are cohesive and whose names reflect the contents of those stores. Figure 16 shows two Business Events with their required, cohesive essential stores.

In this example, the “Customers” store contains only customer information about a Customer; the “Materials” store contains only information about the materials; and the “Customer Invoices” store contains only information about “Customer Invoices.”

Please note that the two Business Events shown may be merged if we found them to be two fragments of the same Business Event (Design Trap #1). In which case we would go on to question the need for the “Customer Invoices” store between them as it may only be a Convenience Store as discussed in the “Partitioning by Business Events” chapter (unless it is used by some other Business Event Partition). If we did combine these two processes into one Business Event, the “Customer Invoices” store would become a transient data flow. Also

note the stimulating data flow "Customer Payment" would become a response data flow (a "pull"). The Event Context Diagram (and any lower-level subprocesses) for this Business Event would obviously also reflect this combination.

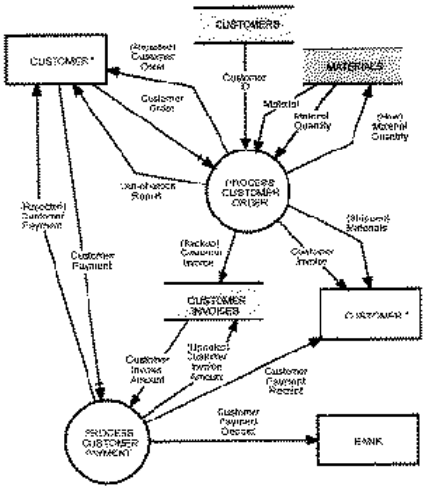


Figure 16 – Essential Data Stores

USING BUSINESS EVENTS TO DRIVE ASSOCIATED REGULATORY AND DEPENDENT EVENTS

If we create a partition and model the "essential" processes and data needed to satisfy a Business Event, this partition would only contain necessary business data and processing. However, even after we've removed all the "system" issues from a Business Event Partition there will still be processing and data needed to support the Business Event. Figure 17 shows these supporting Events with dashed areas representing these issues. The figure shows the four essential processes and their data needed to satisfy our Business Event. It also shows the four support processes and data that aren't directly associated with our Business Event but that are required to satisfy the Dependent and Regulatory Event issues to support our Business Event.

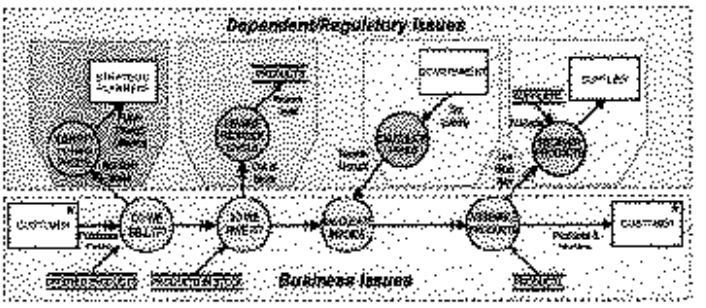


Figure 17 – Business Event with its Supporting Events

If we have the ability to suggest changes to regulatory agencies or make an alliance with our supplier/vendor, then our Business Events can subsume disjointed Regulatory and Dependent Events and/or their aspects.

DYNAMICALLY DEFINED BUSINESS EVENTS

Now that we've talked about the potential problem of historical Events, this leads us to talk about dynamic Events. The designers of old systems typically have not taken a customer "external" view of system partitioning; instead, they took the "internal" system view. These designers tended to look at a customer's need from the point of view of many small requests fragmented across internal systems. Rather than satisfy the customer's one request in full, they expected the customer to interact with the organization's systems to piece together fragmented requests to satisfy their needs.

Sometimes we will find that systems builders have tried to merge two or more of these small, fragmented requests into what the customer may actually want to do in one interaction with the organization.

- A bank deposit with the "less cash" option is an example of this merging: to get a significant complete task done, the bank's customer usually has to perform an interactive dialogue (with either a human or an automatic teller). The dialogue is often interrupted by a number of returns to a menu screen, or there are new forms to fill out before his or her need is met.

The ultimate in customer satisfaction would be for the bank to find out exactly what the customer needed to do and then to perform the entire transaction seamlessly i.e., to create "customized" Business Events. In fact, the term, "customized Business Event" is redundant. If a Business Event isn't fulfilling one complete need of the customer, it's not a properly partitioned Business Event (see Figure 18).

For quality customer satisfaction, we need to identify each specific need of each customer. This might seem ridiculous to some veteran computer people who are used to generic transactions in a system. However, for ultimate customer satisfaction, we truly want to have something at the beginnings of our systems that can form a unique Business Event Stimulus each time and bring together whatever data and processing is necessary in our organization to accomplish that Business Event.

We typically can see this implemented in a manual system because human beings can dynamically adjust to what a customer wants (and probably even recognize and remember that customer's past requests). They can bring together the necessary processes and materials to satisfy a particular customer's specific need; we can do this in a new system without requiring the customer to go to other departments, or to fill out many different forms.

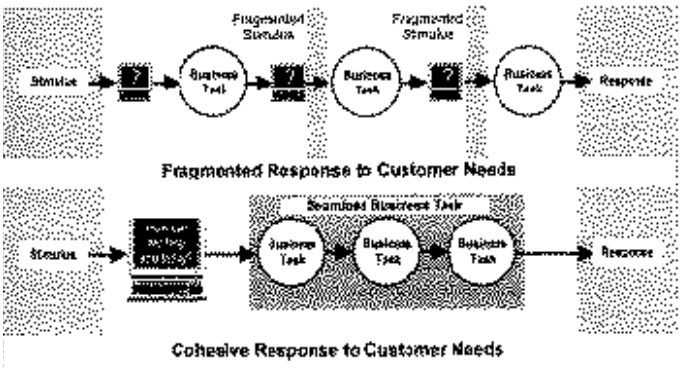


Figure 18 – Business Event Driven Processing

With today's technology we can be far more dynamic and interactive. In fact there is a demand to empower employees with multiple skills, to computer assist employees, and to satisfy the customers' needs with total on-line, real-time computer processing.

In order to provide custom Business Events, we first need to discover what Business Events are typical from the customers' point of view. Once we've identified the processing and the data required to satisfy each of these typical Business Events, then we can satisfy dynamically changing Events by bringing together reusable parts of that processing and data on the fly.

Of course, if the customer requests that we embark on a new Business Event such as new processing and new data, then this involves a Strategic Event and requires us to conduct an analysis.

- In our banking example, the customer may request to buy theater tickets with seat selection. If the Strategic Planners say we wish to satisfy this as-yet undefined Business Event, we analyze our Business Policy for this new Business Event. We can satisfy that

customer's need with the data and processing within our organization's context or we can create them.

This is no different than you having the set of skills and knowledge (processing and data) in a particular subject matter, and having someone request you to do something you've never done before. If you understand the request, you can satisfy that need by pulling together the procedures required to satisfy the demand.

- To beat this to death, let me give an example of a valid bank customer request where: we find out that every other Friday a customer wants to deposit a paycheck; the same time he wants to pay a standard mortgage payment, take out $200.00 in "less cash," put $100.00 in a checking account, and put the rest into a savings account. If we make our customer perform four separate transactions (either in a manual or an automated system), we have not satisfied the customer's needs seamlessly. Now, the customer may accept that that's how it's supposed to be, and he may accept multiple returns to the menu screen, or fill out multiple slips of paper to make his transactions. The sum of these separated transactions is really that customer's Business Event. We need to satisfy this specific need seamlessly before another bank offers this type of new service.

Now this dynamic forming of Business Events may not be easy to implement. We may decide to go back to a menu screen that gives a wide range of customer requests. However, for our Customer Focused Engineering analysis effort we are chartered to produce a customer focused Business Model and not worry about how we are going to ultimately implement it.

For ultimate customer satisfaction, we need to accommodate dynamic Business Events and put together custom Business Events using our system knowledge base to satisfy our customer's needs. Today we can do this bringing together of dynamic Events. For those of you familiar with databases and high-level computer languages (those based on human languages such as English), this dynamic integration is currently available to us. However, this is usually on an ad-hoc basis, recreated each time and requires all the data be available before executing the dynamic request).

When defining a Business Event and its resulting partition our task is to put together that collection of nouns and verbs that satisfy our customer's true needs and this can then be used for new systems design and implementation.

SUMMARY

The foundation of the Business Event Methodology is its focus on the customer and their Events that stimulate our organization into life. Recognizing the five types of Organizational Events that stimulate all organizations and filtering out the Strategic and System Events is another building block of the methodology. It is only by doing this filtering that we can focus on our true business customers (and their complete needs).

EXCELLENCE IN PRACTICE

Remember, at the beginning of this chapter I said Business Model Events were context driven. I hope you can see after reading this definitions chapter why one organization's Business Events will be another organization's System Events.

For example, in most organizations issues to do with computer systems and humans are System Event-related and we remove them to form a valid Business Model. However, if we are in the business of creating computer hardware or providing temporary staffing then these are not System Events. When organizations contract with my own business, Logical Conclusions, Inc., our services are in fact System Event issues to these organizations. However, within the context of my organization, I'm in business to provide training and consulting and therefore, these are my Business Events.

Keep in mind there will be even more traps than I have identified here that are specific to your area of business. I hope the ones we've gone through give you the framework you need to recognize other design traps. Unfragmenting Business Events from old design traps is where a Customer Focused Engineer earns his or her keep. Bringing together Business Events is the most important step in producing a true Business Model.