Relationships between Common Graphical Representations in System Engineering

Jim Long
President
Vitech Corporation
www.vitechcorp.com

Most systems engineers today use graphical representations of a system to communicate its functional and data requirements. The most commonly used representations are the Function Flow Block Diagram (FFBD), Data Flow Diagram (DFD), N2 Chart, IDEF0 Diagram, Use Case, Sequence Diagram, Enhanced Function Flow Block Diagram, and Behavior Diagram. This paper discusses the characteristics of each and shows how they are related.

When analyzed in the context of specifying functional control and data modeling, it appears that the FFBD and DFD representations are limiting, special cases of the Enhanced Function Flow Diagram and Behavior Diagram. The N2 Chart has the same capability as the DFD, with a more formal format. The IDEF0 is essentially an N2 Chart with some control definition (no constructs) capability. The IDEF0 has the capability to indicate the allocation of functions to system components.

Thus, the Enhanced Function Flow Diagram and the Behavior Diagram features comprise a "parent/unified" set of graphical system representations. To achieve the same level of specification completeness, you would have to use an integrated set of the FFBD and one of the data models or augment the FFBD with a graphical representation of the data model, as was done at TRW (then called Function Sequence Diagrams).

Background

Over the past several years, systems engineers have evolved to a few graphical representations to present the functional and data flow characteristics of their system design. The most common of these are the Function Flow Block Diagram (FFBD), Data Flow Diagram (DFD), N2 (N-Squared) Chart, IDEF0 Diagram, Use Case, Sequence Diagram, Enhanced Function Flow Block Diagram, and Behavior Diagram (BD). Most of these graphical representations allow the engineer to decompose the functional and/or data models hierarchically. The objective of this paper is to analyze the representation capability of these graphic "languages" to see if there is a unifying view available.

Terminology

Let us introduce two terms that we use in describing the conditions that allow/cause a function to begin execution. Considering the control and data environment, a function can begin execution if it is both enabled (by control) and triggered (by data). In the case where there is no data trigger specified, a function begins execution upon being enabled. A function is enabled if the function(s) that precede it in the control flow specification have completed execution (e.g., satisfied their completion criteria). A function is triggered when the required stimulus data item becomes available to the function. We are not concerned here with other execution requirements (such as the availability of necessary resources, except in the case of the Dynamic Timeline Diagrams) that could be represented by either control or data structures as necessary.

Copyright © 2002 Vitech Corporation. All rights reserved.

Function Flow Block Diagram

The Function Flow Block Diagram (FFBD) was the first to be favored by systems engineers and continues to be widely used today (DSMC 1989, Blanchard and Fabrycky 1990). Figure 1 shows a sample FFBD. An FFBD shows the functions that a system is to perform and the order in which they are to be enabled (and performed). The order of performance is specified from the set of available control constructs shown in Figure 2. The control enablement of the first function is shown by the reference node(s) which precede it, and the reference node(s) at the end of the function logic indicate what functions are enabled next. The FFBD also shows completion criterion for functions as needed for specification (for example, the exits for the multi-exit function in Figure 1). The FFBD does not contain any information relating to the flow of data between functions, and therefore does not represent any data triggering of functions. The FFBD only presents the control sequencing for the functions.

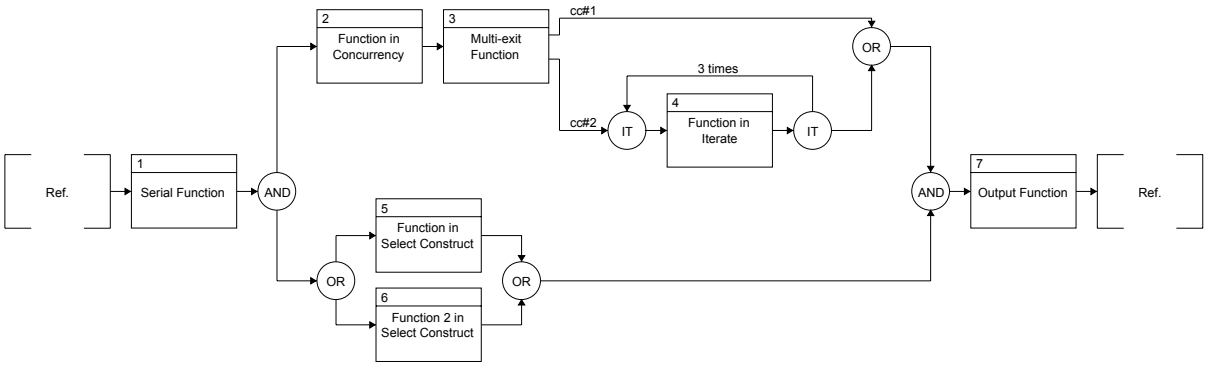

Figure 1 – Sample Function Flow Block Diagram (FFBD)

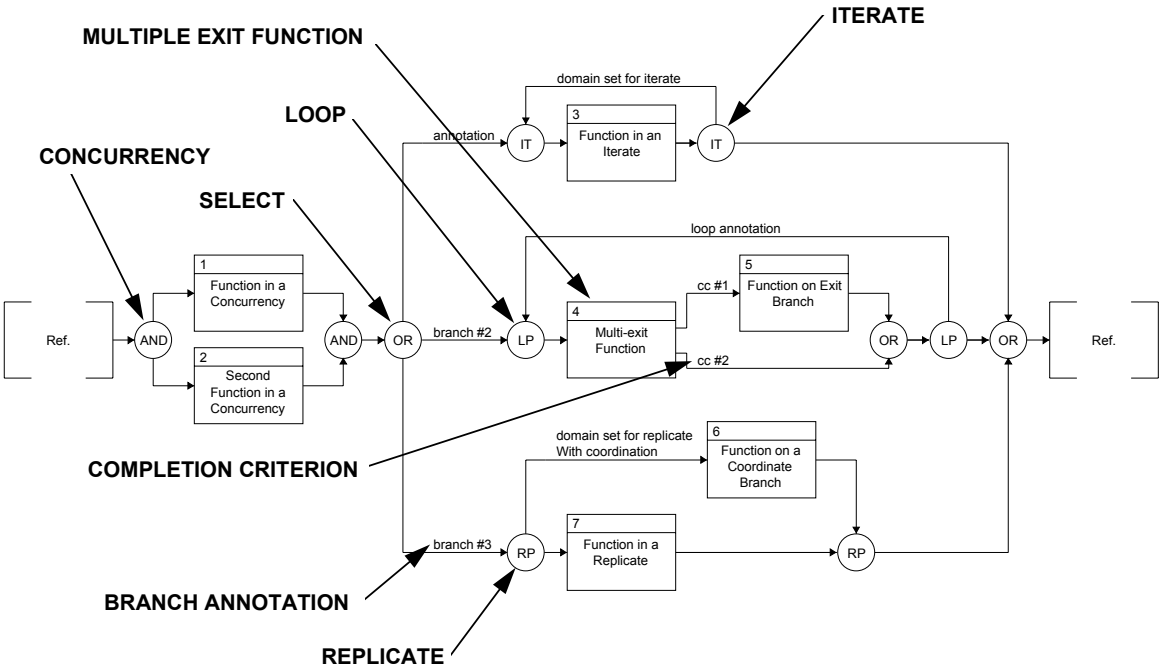

Figure 2 – Control Constructs Available for Function Flow Block Diagrams

Copyright © 2002 Vitech Corporation. All rights reserved.

Data Flow Diagram

The Data Flow Diagram (DFD), shown in Figure 3, shows required data flow between the functions of a system (DeMarco 1979). This representation has been widely used by software engineers and serves as the basis of many software engineering methodologies and automated tools. The figure shows that data repositories, external sources, and external sinks can also be represented by DFDs. However, DFDs do not show any control constructs for function sequencing or enablement.

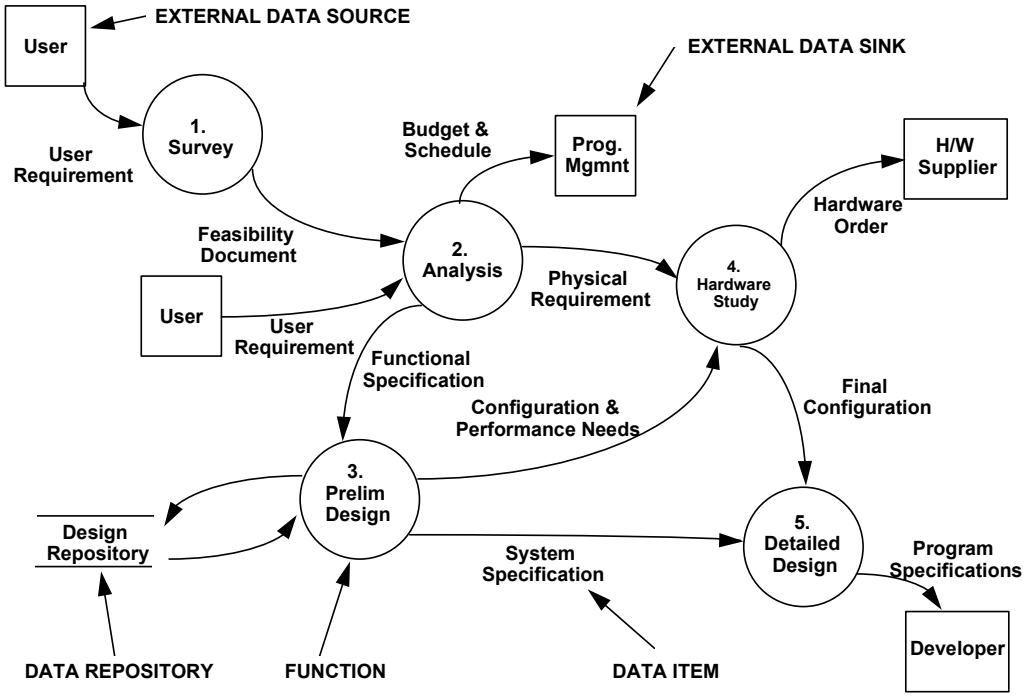

Figure 3 – Sample Data Flow Diagram

N2 Chart

The N-Squared (N2) Chart, shown in Figure 4, was developed to show and specify interfaces between the elements of a system (Long et al. 1968, Lano 1977). Figure 4 is the N2 Chart that corresponds to the Enhanced FFBD in Figure 8. When used to show the interfaces between the functions in a system, the N2 chart is equivalent to a DFD — it contains all the information and differs only in format. The N2 chart is commonly used as a complement to the FFBD to provide the data flow information as inputs and outputs of the system functions.

The N2 Chart is structured by locating the functions on the diagonal, resulting in an N x N matrix for a set of N functions. For a given function, all outputs are located in the row of that function and all inputs are in the column of the function. If the functions are placed on the diagonal in the nominal order of execution, then data items located above the diagonal represent normal flowdown of data. Data items below the diagonal represent data item feedback. External inputs can optionally be shown in the row above the first function on the diagonal, and external outputs can be shown in the right-hand column. If desired, data repositories can be represented by placing them on the diagonal with the functions.

Copyright © 2002 Vitech Corporation. All rights reserved.

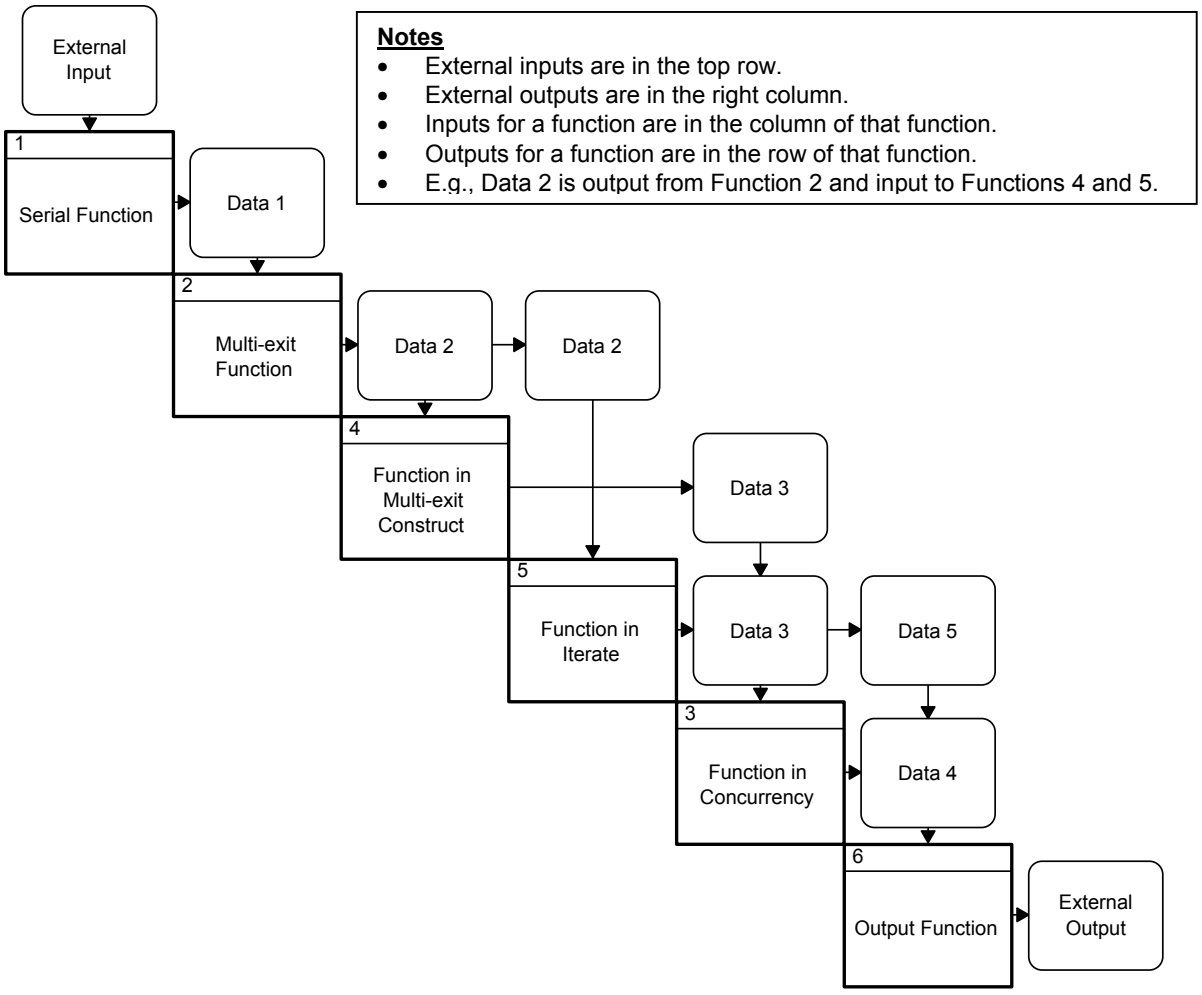

Figure 4 – Sample N-squared (N2) Chart

IDEF0 Diagram

The IDEF0 Diagram (see Figure 5) appears to be a derivative of the DFD with a format like the N2 Chart (Groveston, 1989 and FIPS-183, 1993). The primary content of the IDEF0 Diagram is the specification of data flow between system functions. An IDEF0 diagram is often supported by an IDEF1 or IDEF1x diagram indicating data keys with the objective of preserving referential integrity. The IDEF0 diagram does allow the specification of control as an input to a function but does not have the capability to characterize that control in terms of constructs, as the FFBD, EFFBD, and Behavior Diagrams do. The specification of control with the IDEF0 notation is incomplete and, therefore, not executable. The IDEF0 Diagram also represents the mechanism (usually the component to which the function is allocated) which performs the function. Figure 5 is the IDEF0 Diagram that corresponds to the EFFBD in Figure 8.

Use Case Diagram

The Use Case diagram, shown in Figure 6, was primarily created to show human initiated functionality so it doesn't provide all the functions needed in interface management and/or scenario definition and analysis. This representation is widely used by software engineers.

Copyright © 2002 Vitech Corporation. All rights reserved.

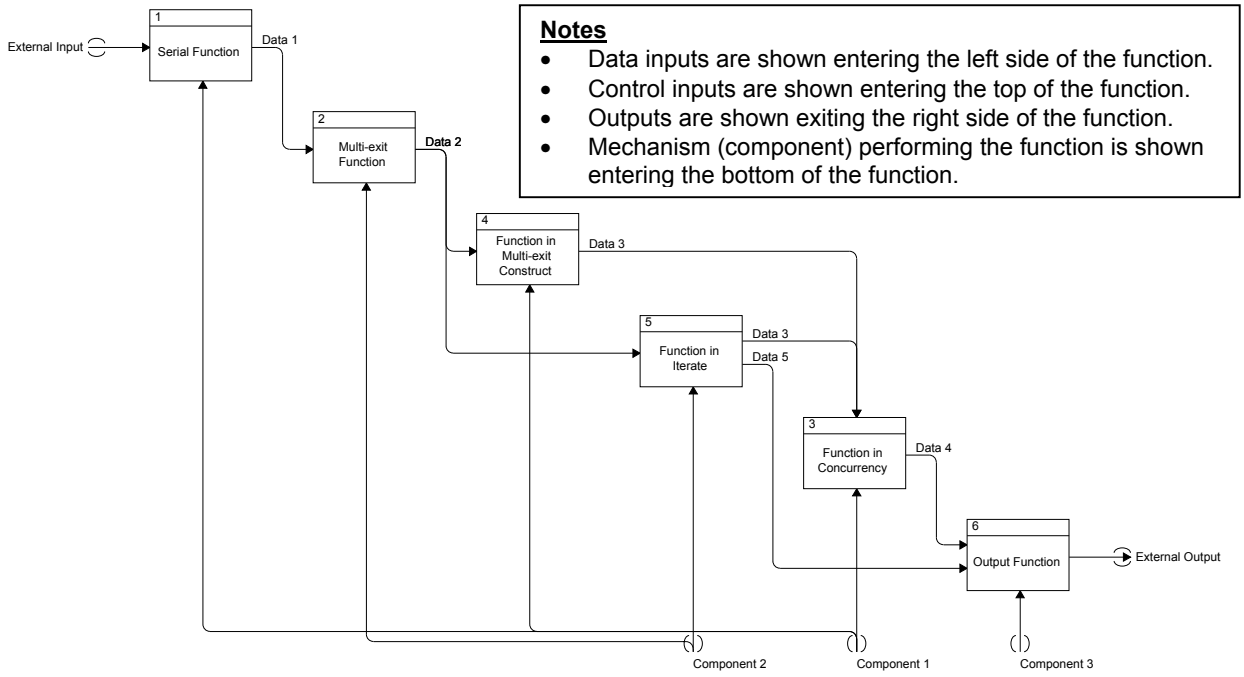

Figure 5 – Sample IDEF0 Diagram

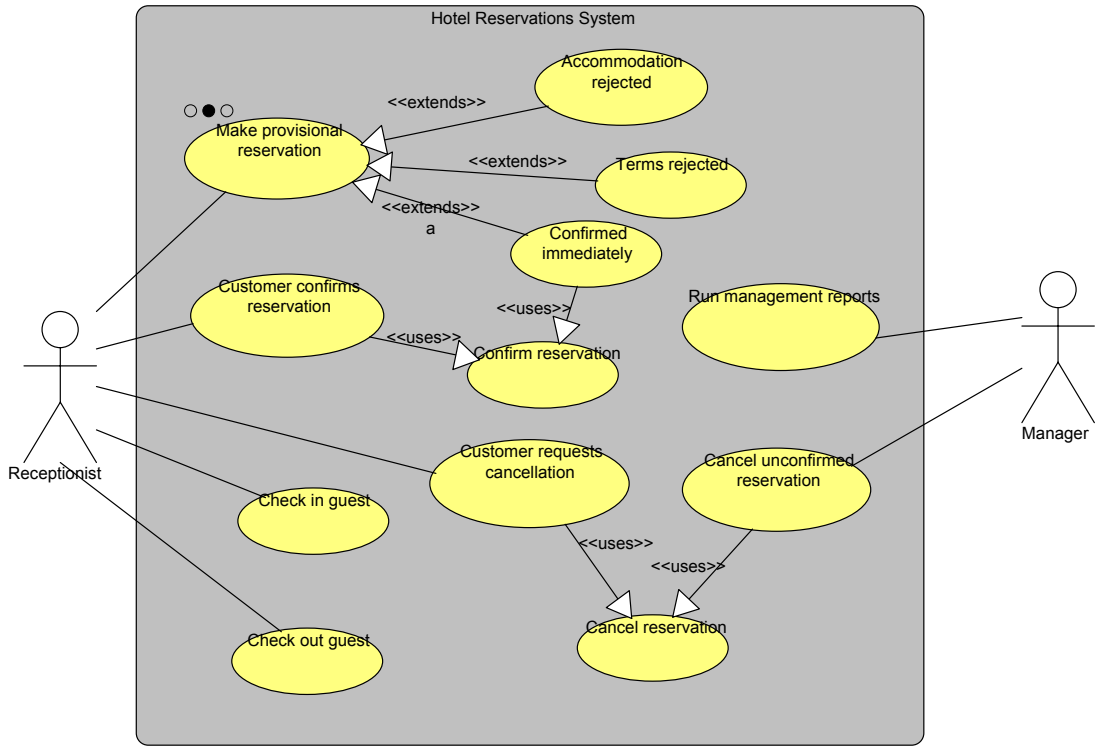

Figure 6 – Sample Use Case Diagram

Copyright © 2002 Vitech Corporation. All rights reserved.

Sequence Diagram

Sequence diagrams, as shown in Figure 7, describe interactions among physical components in terms of an exchange of messages over time. The primary content of the sequence diagram is the specification of data flow between a subset of system components. The sequence diagram does not have the capability to characterize control in terms of constructs, as the FFBD, EFFBD, and Behavior Diagrams do. The specification of control with the sequence diagram notation is incomplete and, therefore, not executable. This representation is widely used by software engineers.

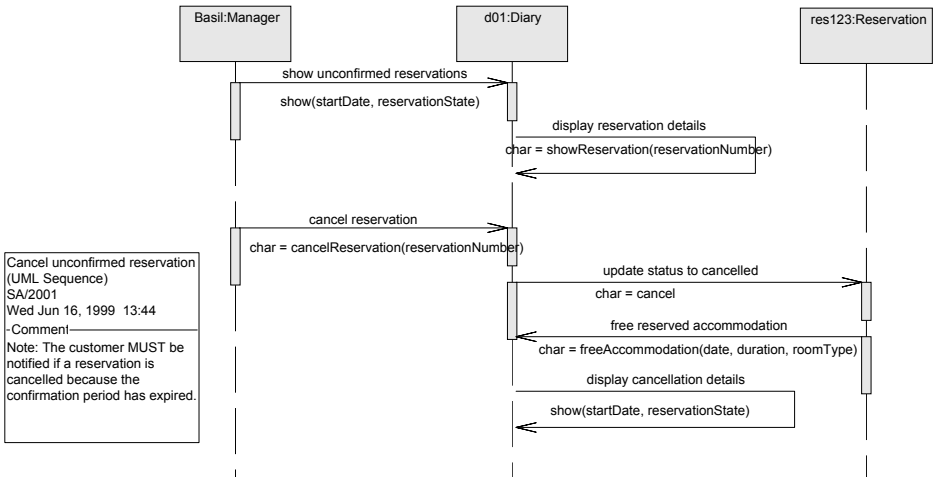

Figure 7 – Sample Sequence Diagram

Enhanced FFBD

The EFFBD displays the control dimension of the functional model in an FFBD format with a data flow overlay to effectively capture data dependencies. Thus, the Enhanced FFBD represents: (1) functions, (2) control flows, and (3) data flows. The logic constructs allow you to indicate the control structure and sequencing relationships of all functions accomplished by the system being analyzed and specified. When displaying the data flow as an overlay on the control flow, the EFFBD graphically distinguishes between triggering and non-triggering data inputs. Triggering data is required before a function can begin execution. Therefore, triggers are actually data items with control implications. In Figure 8, triggers are shown with green backgrounds and with the double-headed arrows. Non-triggering data inputs are shown with gray backgrounds and with single-headed arrows.

The Enhanced FFBD specification of a system is complete enough that it is executable as a discrete event model, providing the capability of dynamic, as well as static, validation. A fundamental rule in the interpretation of an EFFBD specification is that a function must be enabled (by completion of the function(s) preceding it in the control construct) and triggered (if any data input to it is identified as a trigger) before it can execute. This allows the engineer maximum freedom to use either control constructs or data triggers (or a combination of both) to specify execution conditions for individual system functions. By augmenting the EFFBD with function duration estimates/budgets and resource constraints and utilization, trade studies can be done using dynamic simulation outputs. Figure 9 shows the output of such a simulation. This figure shows timelines for a resource (MIPS). It also shows timelines for the integrated system functions, indicating the delays in function execution due to waiting for triggers and resources (MIPS). This dynamic view is also capable of showing the status of the queuing of triggering data.

Copyright © 2002 Vitech Corporation. All rights reserved.

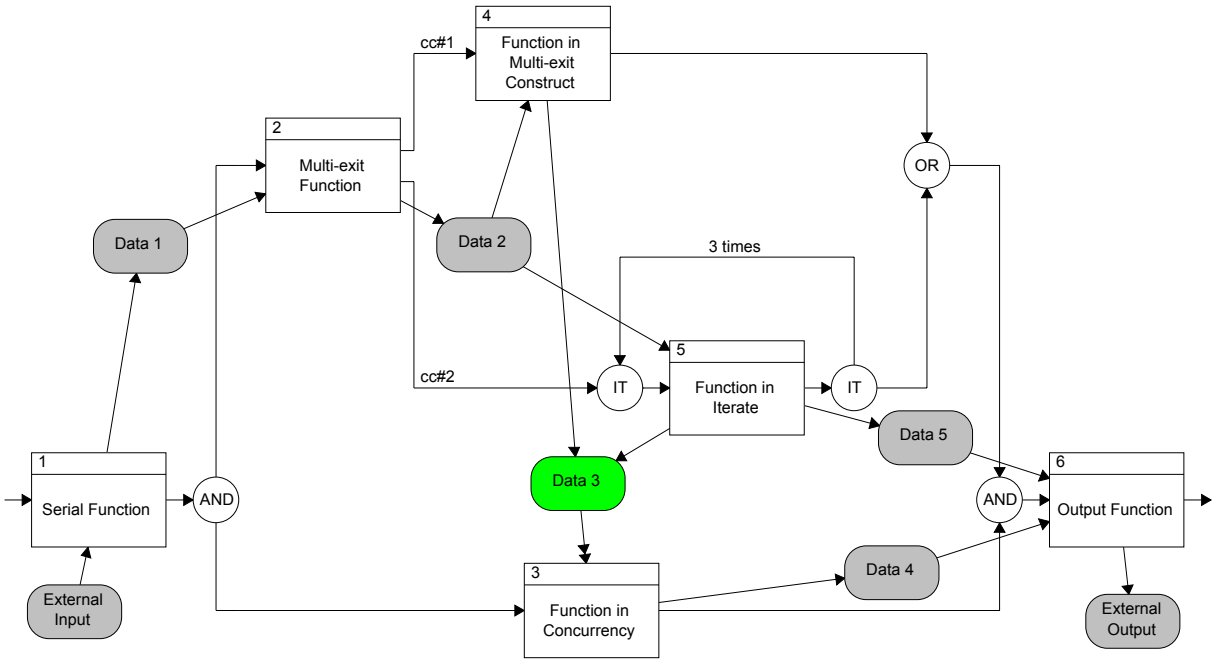

Figure 8 – Sample Enhanced FFBD

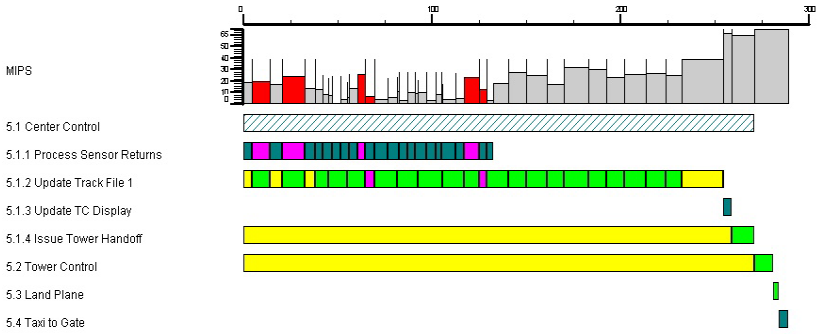

Figure 9 – Dynamic Timeline

Behavior Diagram

The Behavior Diagram (BD) (Figure 10) is a graphical representation equivalent to the Enhanced FFBD (Figure 8). The primary difference is in the orientation of the control flow: in the EFFBD, control sequencing is from left to right; in the BD, control sequencing is from top to bottom. While it is not shown on the graphical construct, the BD model allows data inputs to a function to be characterized as either triggering (a control capability) or data update (not a control implementation).

As with the Enhanced FFBD, the Behavior Diagram specification of a system is sufficient to form an executable model allowing dynamic validation via discrete event simulation methods.

Copyright © 2002 Vitech Corporation. All rights reserved.

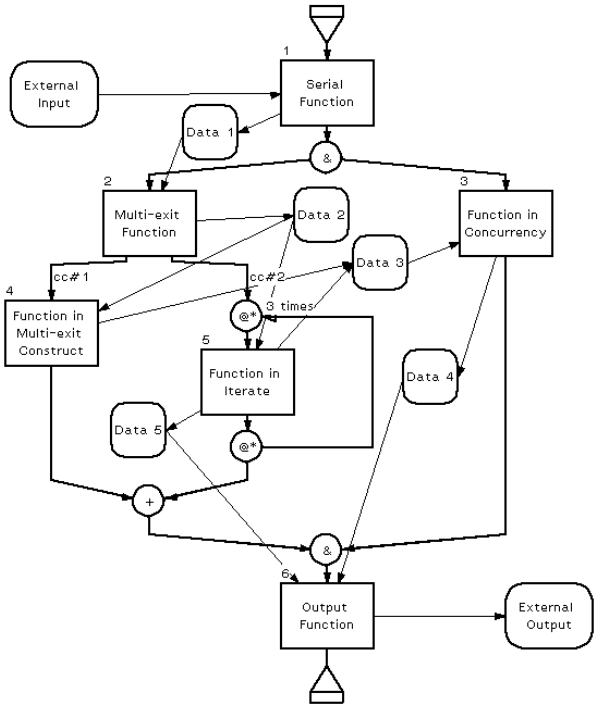

Figure 10 – Sample Behavior Diagram

Relationships between the Diagrams

From the descriptions of the graphic representations, it is seen that the FFBD characterizes only the functional control model and the DFD characterizes only the functional data model for a system. The Enhanced FFBD and Behavior Diagram capture both these limiting models and the continuum between them. Figure 11 is drawn to show the characteristics of the behavior spectrum from one limiting case to the other. Figure 12 shows where all these graphical representations fit on the behavior spectrum and, therefore, places them in perspective relative to each other in the sense of specifying control and triggering for functions.

Conclusions

The graphical representations that systems engineers commonly use to describe and specify the functionality and data requirements of a system are very closely related when analyzed in the context of data and functional control capabilities. In particular, the FFBD and DFD are limiting cases of the EFFBD/Behavior Diagram representation. The N2 Chart is equivalent to the DFD: so it is, likewise, a limiting case of the EFFBD/BD on the data modeling end of the spectrum. The IDEF0 is essentially a DFD, except that some control capability (no control constructs) is added. The IDEF0 also allows the explicit representation of functional allocation (i.e., what system component performs each function).

Copyright © 2002 Vitech Corporation. All rights reserved.

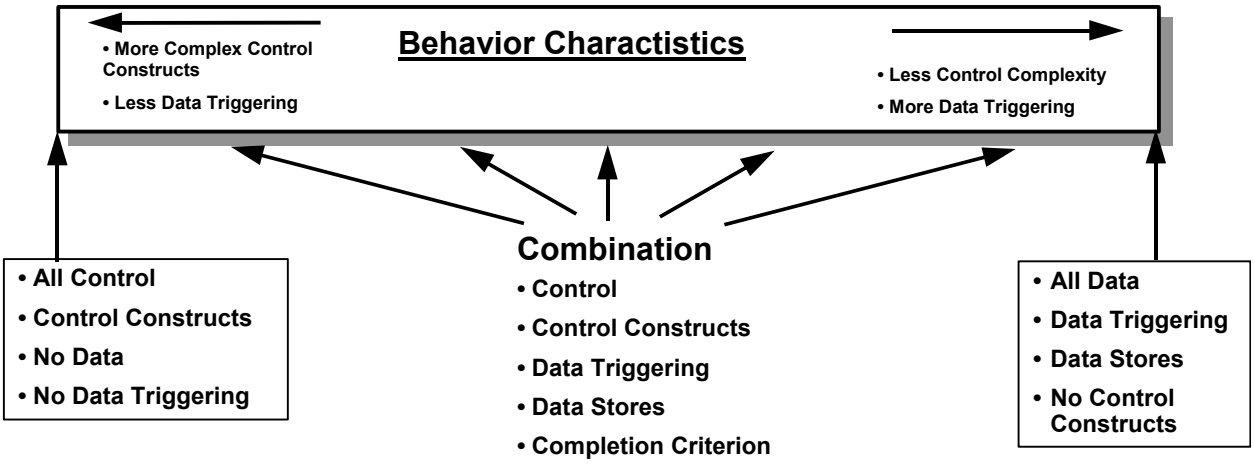

Figure 11 – The Control Enablement and Data Triggering Spectrum

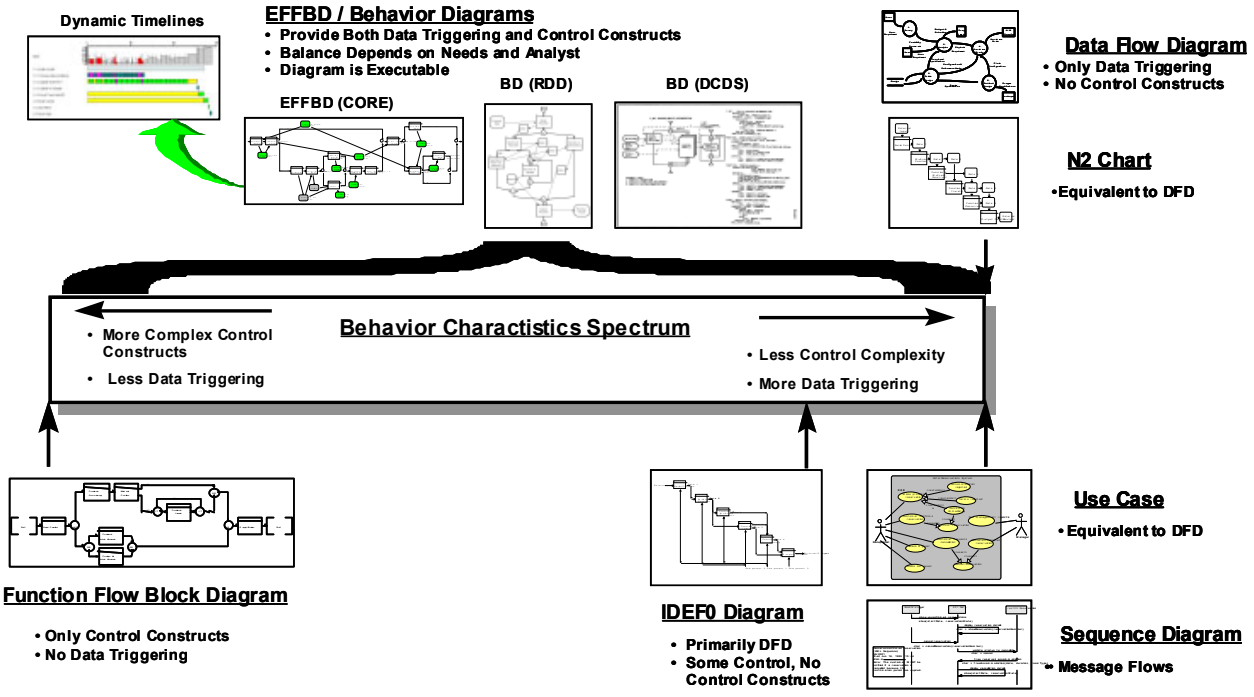

Figure 12 – Relationships of All Graphical Representations – FFBDs and DFDs Are Limiting Cases

Copyright © 2002 Vitech Corporation. All rights reserved.

Enhanced FFBDs and CORE provide the systems engineer with capabilities to perform user interface and control system tasks that are simply not supported by data-oriented diagrams (i.e., those on the right side of the continuum). Specifically:

- The systems engineer uses EFFBDs to represent a high degree of interaction between the user and the system being designed.
- The system engineer uses EFFBDs to include various system-generated interrupts, alerts, and notifications that span the control systems design and the user interface.
- The systems engineer uses EFFBDs to integrate the design of the control system with the operations system. Conditions such as (1) detecting anomalies, (2) making decisions such as continuing to operate in a degraded mode (according to previously established criteria) and (3) returning to normal operation.

The concurrent development of the control system, operations system, and user interface is something that cannot be separated and can never be retrofitted.

Thus, the EFFBD features comprise a "parent" or unifying set of graphical system representations. To achieve the same level of specification completeness, you would have to use an integrated set of the FFBD and one of the data models or augment the FFBD with a graphical representation of the data model, as was done in the 1960s at TRW (where they were called Function Sequence Diagrams).

As stated earlier, a fundamental rule in the interpretation of an EFFBD specification is that a function must be enabled (by completion of the function(s) preceding it in the control construct) and triggered (if any data input to it is identified as a trigger) before it can execute. This allows the engineer maximum freedom to use either control constructs or data triggers (or a combination of both) to specify execution conditions for individual system functions. It also means that if a systems engineer draws EFFBDs with minimal use of data triggering, the engineer is essentially generating an FFBD. If the EFFBD makes minimal use of the control constructs (the structure is mostly parallel/concurrent), the engineer is generating a DFD representation of the system.

About the Author

Mr. James Long is the President of Vitech Corporation – the developer of the systems engineering support tool CORE. He has been a performing systems engineer and innovator since creating the first integrated functional control-data flow diagrams (then called Function Sequence Diagrams) at TRW in 1967. He played a key technical and management role in the maturing and application of that systems engineering process and technology at TRW and Vitech. He is a member of INCOSE and served as president of the Washington Metropolitan Area Chapter, the largest chapter of INCOSE.

References

Blanchard, B. and Fabrycky, W., *System Engineering and Analysis*, Prentice Hall, 1990.

Defense Systems Management College, *Systems Engineering Management Guide*, U.S. Government Printing Office, 1989.

DeMarco, T., *Structured Analysis and System Specification*, Prentice Hall, 1979.

FIPS-183, Draft Federal Information Processing Standards Publication 183, NIST, 1993

Groveston, P. K., *Functional Diagramming Methods*, Unpublished MITRE Briefing, March 1989.

Lano, R., *The N2 Chart*, TRW Software Series, 1977.

Long, Dinsmore, Spadaro, Alford, et al., *The Engagement Logic and Control Methodology as Derived, Defined, and Applied at TRW*, unpublished notes, 1968 - 1972.

Oliver, D., *Systems Engineering and Object Technology*, Proceedings of the Fourth Annual International Symposium of NCOSE, August 1994.

Originally published in the Proceedings for the 1995 NCOSE Symposium.
Updated July 2002

Copyright © 2002 Vitech Corporation. All rights reserved.