Role of System Engineering Across The System Life Cycle

Loyd Baker, Jr.
James E. Long
Vitech Corporation
2070 Chain Bridge Road, Suite 105
Vienna, Virginia 22182-2536

Abstract. The role of the system engineer does not end upon the publication of the system-level system specification, e.g., SSS, IRD, etc. In fact, the system engineer's role does not end until the system has been decommissioned and discarded.

BACKGROUND

While many of our system/process development projects have been document-centered, the experienced system engineers individually build *models* to better understand the problem to be solved, develop candidate solutions, trace related pieces of information, and validate their decisions. However, these knowledge-rich engineering models developed on an adhoc, informal basis, were not formally retained as part of the design/planning process, therefore they were not available to support down-stream development/test activities, or future operation and maintenance changes. Therefore, costs and schedules were impacted because the project had to re-understand what had previously been discovered, and subsequently forgotten as a result of personnel changes.

Recognizing the need to formalize the emerging model-driven system engineering approach, an INCOSE Interest Group was formed in 1995 to characterize Model Driven System Design (MDSD) [1]. This paper discusses the concept using an existent, proven, model-oriented system engineering process developed by TRW for the U.S. Army, Ballistic Missile Defense Advanced Technology Center, and productized by Vitech Corporation, the developer of the PC-based system engineering support tool, CORE®.

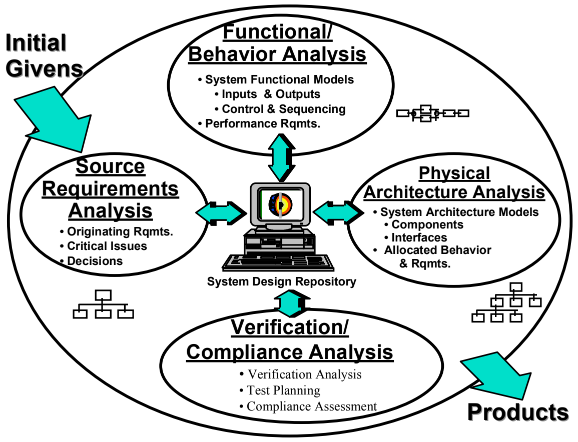


Figure 1 - System Logic Modeling (SLM) Process

The following kinds of models are used to help focus the system/business process designer on the appropriate set of questions that need answering in order to find the most reliable and cost effective solutions.

- **Schematic Model**: A chart or diagram showing object inter-relationships, parent-child structures, and time sequencing of actions (e.g., organizational chart, parts-list, traceability model, sequence diagram).
- **Physical model**: Geometric equivalents used for reality experimentation and demonstration (e.g., DNA model, model of the solar system).
- **Mathematical Model**: Coded instructions which emulate environmental phenomena, system/process behavior, or monitor conditions and make predictions.

The four major activities identified in the SLM process are applied at each level of system specification/design. First, the givens (i.e., relavent text/model information used as a starting point) are reviewed to understand the problem to be solved, and capture all previously defined requirements and design constraints. This set of source requirements are analyzed for fesability, completeness, consistency, and testability. Identified issues and risks are documented for futher analysis, with the resolution captured as decisions. The resulting **Source Requirements Analysis Model** is the basis for the three concurrent/iterative modeling activities presented in Figure 1: Functional/Behavior Analysis, Physical Architecture Analysis, and Verification & Compliance Analysis. These concurrent activities identify the physical component parts of the system, the functional sequence flows that model the desired behavior of the system and its component parts, and the verification / test plans used to perform requirements compliance assessment.

INTRODUCTION

In today's global marketplace, customers everywhere expect total quality from an organization's products and services. Organizations that plan on achieving, and maintaining, a competitive edge are actively re-engineering their organizations and processes. Corporations are changing their product/service organizations from a "task-oriented" structure to a "information modeling oriented" structure. For example, rather than stating requirements, proposed solutions, and verification plans as a hierarchical collections of English text that can only be evaluated/analyzed by human inspection, the information is being expressed as a collection of integrated information models. These models, used to understand and specify a system/process, must be human readable to enhance communication of intentions between team members and the customer, and the models must be machine executable such that the dynamics of the system/process can be automatically analyzed early in the development process. Additionally, these machine executable information models must support the automatic generation of review documents and formal specification documents. Automatic generation of documents from the engineering database eliminates, or at least greatly reduces, the need to manually build project review material and specifications independently from the engineering database where the majority of the data/information being reported already exists.

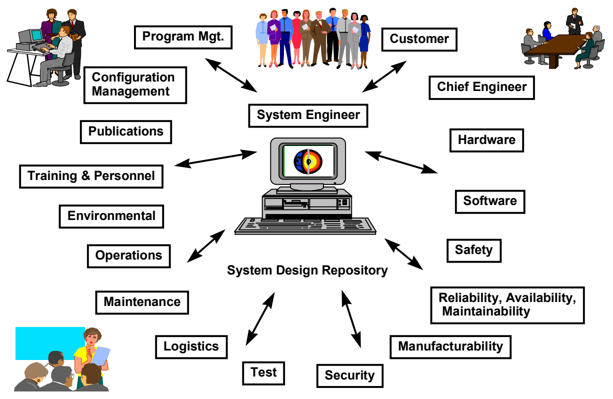
This paper discusses a proven, flexible, yet very simple to understand and use, model-driven approach based on the **entity-relationship-attribute plus flows (ERA+Flows)** information modeling specification language. This specification language supports the project engineers efforts to identify the *things* of importance in the system/process (i.e., an entity is a *thing*), the characteristics/properties (i.e., attributes) of those *things*, the relationship between *things*, and the internal behavior (i.e., conditional process-flow and item-flow models) of the *things* being specified. This information modeling technique serves as an artificial language (i.e., structured text and graphical constructs) in which the problem, candidate solutions, and verification approaches can be understood and communicated to all the developers, testers, and potential users of the system/business process. This specification language is used to implement the SLM System Engineering Process presented in Figure 1, and described in the following sections.

SPECIFICATION LANGUAGE OVERVIEW

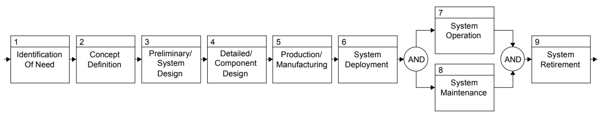
The Information Modeling Specification Language capable of describing an evolving system/business process design is implemented using an entity-relationship-attribute (ERA) plus flows language. The ERA+Flows language building blocks, described in [2], are summarized below:

- **Entities** (i.e., objects) identify the *things* of importance in an organization, system, or process. Entities are the objects that serve as the basic units of knowledge in the SLM System Engineering Process. An entity corresponds to a noun in English.

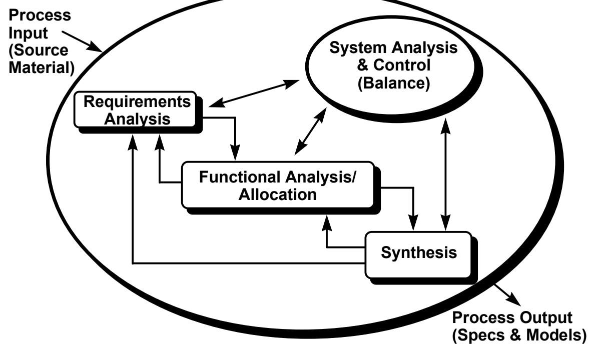
- **Relationships** define associations between the system/process entities. Relationships are similar to verbs (e.g., RawMaterials "*consists of*" Fruit). The labeled arcs, connecting the entities are the relationships. Relationships can have attributes.



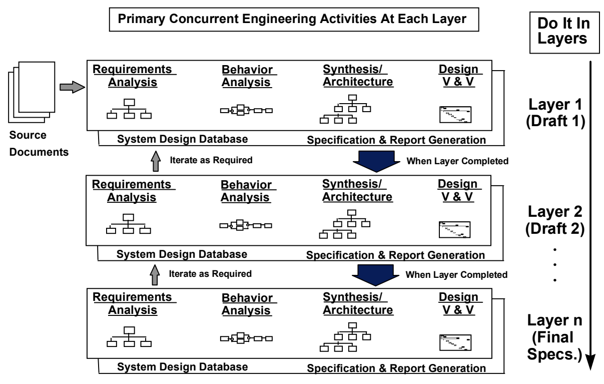
- **Attributes** define the properties/characteristics of an entity/relationship much like adjectives modify nouns. For instance, attributes of a Requirement entity might include the requirement name, requirement statement, rationale, and status.



- **Flows** define conditional sequencing of activities associated with a system/process entity, and the flow of interfacing items passing into and/or out of the activities being modeled. A process-flow (i.e., work-flow), graphical model, consists of time precedence action-nodes (rectangles) and control-nodes (circles) used to model sequential, parallel, conditional, and iterative processing. In the SLM methodology, the process-flow model is implemented as a Function Flow Block Diagram (FFBD). The flow of interfacing items (things, data) associated with the process-flow activities are presented in the form of an N-Squared (N2) Diagram. These two separate views are used to enhance readability in complex models. The SLM system engineering support tool constructs the different views from a single integrated work-flow (i.e., behavior) model stored in the System Design Repository. A complete description of the flow modeling techniques can be found in [3]. Example process-flows and item-flows are as follows. In the following example the processing to be associated with function "harvest fruit" is shown as an FFBD and the item-flow as an N2 Diagram.


Process Input (Source Material)
System Analysis & Control (Balance)
Requirements Analysis
Functional Analysis/ Allocation
Synthesis
Process Output (Specs & Models)

In summary, the ERA+Flows constructs form an artificial language that defines and documents the precise meaning of each concept, serving to clarify and enhance communications between system developers and users. These models are used at all levels of system/process specification (i.e., requirements definition, design, and verification planning) to represent "what" activities must be performed, "when" they should be performed, and "how well" they should be performed. These models form the basis for "executable specifications" needed to support dynamic analysis of the evolving business process/system design. Additionally, the ERA constructs provide



- **checklist** of information which needs to be specified for the system specification to be complete.
- **roadmap** used to automatically generate required documents directly from the evolving engineering database, reducing the time required to publish project review material, and deliverable formal specifications.

The following sections discuss the construction and use of different kinds of information models used to support the SLM model-driven system engineering approach. A "book publishing process", for the Ajax Publishing House (APH), will be used to illustrate the model-driven concepts.

SOURCE REQUIREMENTS ANALYSIS MODEL

The initial activity to be performed consists of identifying all the relevant source material from which our set of "starting point" (i.e., source) requirements can be extracted, categorized, and recorded in the information repository. The hierarchical structure (i.e., parent-child relationships) of the extracted requirements are captured in order to track changes necessitated by future updates to the relevant set of source documents. This "requirements traceability model" expands as new "derived requirements", "issues", and "risks" are added to the model. The leaf level requirements in the traceability model should be single, testable requirement statements. Figure 2 illustrates the corresponding information model consisting of entities, attributes, and relationships between entities.

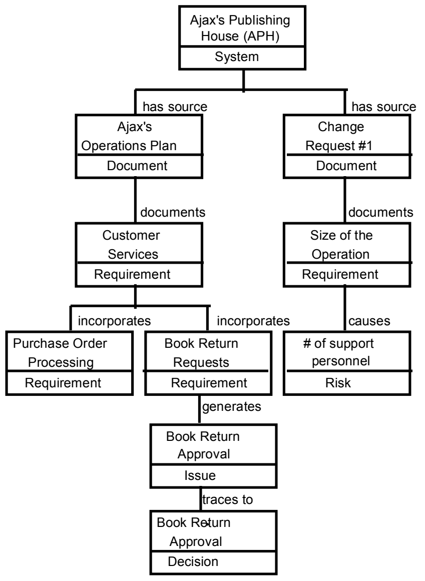


Figure 2 - Requirements Traceability Model

Next, using the knowledge gained from constructing the Requirements Traceability Model, the external interfaces to/from our Ajax Publishing House are derived. The physical entities (i.e., Book Customers, Book Authors, Printers, Ajax Publishing House) and the interfacing items (e.g., Purchase Order) are shown in Figure 3.

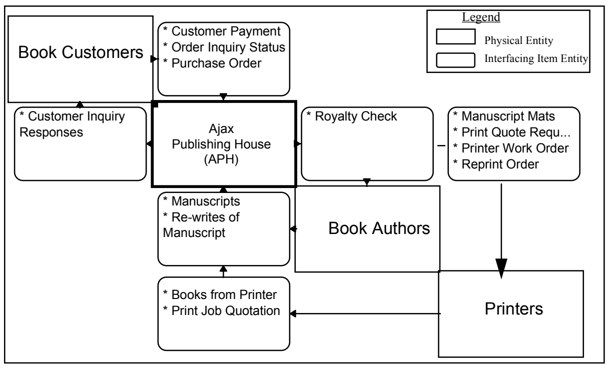


Figure 3 - External Interface Model

This External Interface Model defines the context in which our system/process must operate, satisfying the requirements defined in the Requirements Traceability Model. The specification of the system's Physical Architecture is accomplished in the following steps:

- Define the system's *internal* component parts model (i.e., "parts-list").
- Define the system's *internal* Functional/Performance Behavior Model (i.e., process-flows, item-flows, and performance measures).
- Allocate individual paths of our Functional/Performance Behavior Model onto the system's component parts. This allocation process serves two purposes: 1) Provides each component with the necessary "smarts" to carry out it's desired mission; and 2) Exposes the *interfacing items* (i.e., information, objects) that must be transferred between the system's component parts.
- Define the *physical transfer links* that will "carry" the *interfacing items* between the system's component parts.

INTERNAL COMPONENT PARTS MODEL

The Component entity-type and the *built from* relationships are used to show the parent-to-child association of the system's component parts. Figure 4 presents APH's "parts-list".

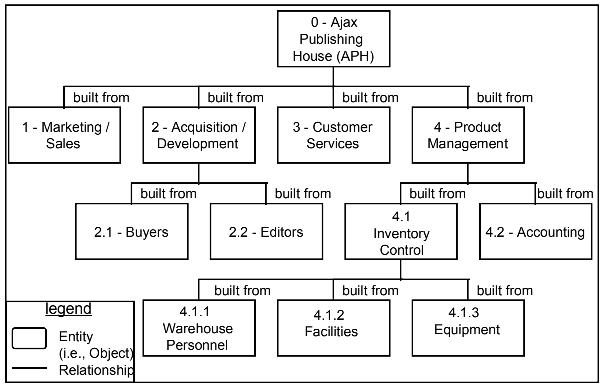
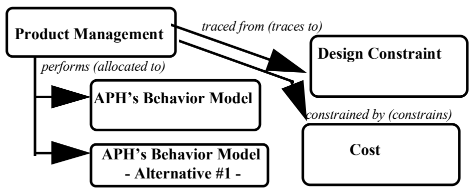


Figure 4 - Internal Component Parts Model

Through out the system life-cycle, additional knowledge is associated with the "parts-list" model until a complete ERA+Flows specification is developed for the component parts of the system (i.e., the architecture is described).



FUNCTIONAL/PERFORMANCE BEHAVIOR MODELS

The process-flow/item-flow structures of the ERA+Flows Specification Language are used to specify the stimulus-response processing paths (i.e., conditional sequences of activities) to be performed by the "parts-list" components of the system. A process-flow model consists of time precedence activity-nodes and control-nodes which indicate concurrence (parallel processing), conditional branching, and sequential iteration loops. The process-flow/item-flow gives us the ability to model candidate solutions to understand, and better communicate to the development team, and customers, the required order of each activity, and the required functional interfaces. An example APH process-flow is shown in Figure 5.

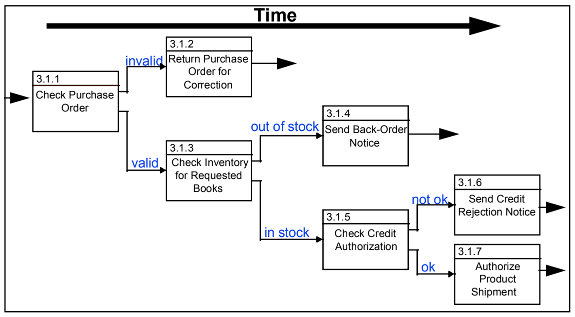
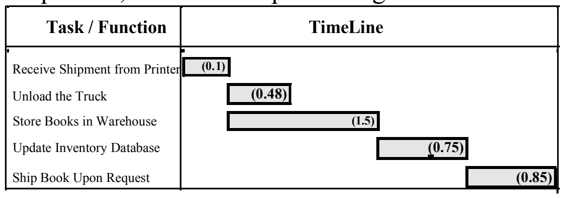

Time
3.1.1 Check Purchase Order
invalid
3.1.2 Return Purchase Order for Correction
valid
3.1.3 Check Inventory for Requested Books
out of stock
3.1.4 Send Back-Order Notice
in stock
3.1.5 Check Credit Authorization
not ok
3.1.6 Send Credit Rejection Notice
ok
3.1.7 Authorize Product Shipment

Figure 5 - Purchase Order Process-Flow Model

Specifying functional/performance requirements and/or design solutions as process-flow models makes <u>explicit</u> the sequences of required processing. It also provides for direct evaluation/verification of our proposed problem solutions because the models are executable. Our system engineering tool-set uses a discrete-event simulation framework to better understand the dynamic characteristics of the system /business process-flow models. The *executable framework* provides the following kinds of dynamic analyses, especially exploring situations that change over time.

- Verification of timing consistency in a process-flow model containing any combination of concurrent, sequential, and iterative processing.



- Verification of input/output consistency.
- Verification of resource management budgets.
- Verification that assigned performance mearsures are within specified limits.

The following example (see Figure 6) illustrates instrumenting a process-flow model to identify performance measures defined to better track system risks.

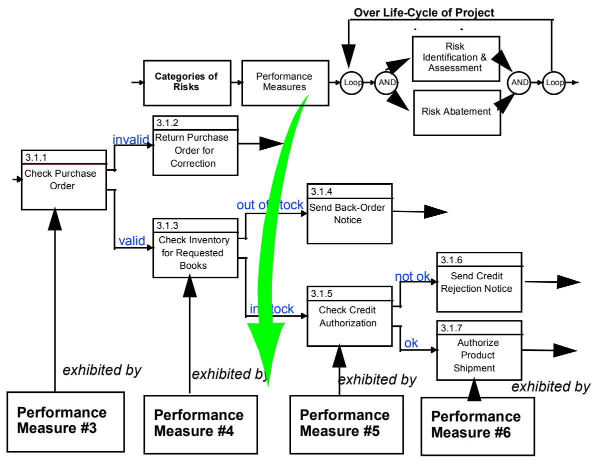

Over Life-Cycle of Project
Categories of Risks
Performance Measures
Loop
AND
Risk Identification & Assessment
Risk Abatement
AND
Loop
3.1.1 Check Purchase Order
invalid
3.1.2 Return Purchase Order for Correction
valid
3.1.3 Check Inventory for Requested Books
out of stock
3.1.4 Send Back-Order Notice
in stock
3.1.5 Check Credit Authorization
not ok
3.1.6 Send Credit Rejection Notice
ok
3.1.7 Authorize Product Shipment
exhibited by
exhibited by
exhibited by
exhibited by
Performance Measure #3
Performance Measure #4
Performance Measure #5
Performance Measure #6

Figure 6- Assignment of Performance Measures

PHYSICAL ARCHITECTURE MODEL

Once the system's Behavior Model is verified, it is partitioned and allocated to the system's component parts (i.e., subsystems).

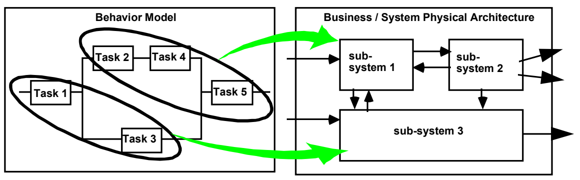


Figure 7 - Allocation of Behavior to Physical Architecture Model

The allocation process exposes the *interfacing items* (e.g., Credit Inquiry) flowing between the system's component parts. The physical transfer mechanisms (i.e., modem, phone, etc.) to be used to accomplish the identified functional interfaces are specified using the Link entity-type. A physical Link "carries" the *interfacing items* (see Figure 8).

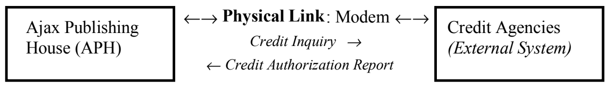

Ajax Publishing House (APH)
←→ Physical Link: Modem ←→
Credit Inquiry →
← Credit Authorization Report
Credit Agencies
(External System)

Figure 8 - Physical Interface Links

Finally, the completed Physical Architecture Model must be analyzed for feasibility and compliance to the Requirements Traceability Model. The planning for this verification activity is conducted using the same ERA+Flows Specification Language, but augmented with verification planning specific entities, attributes, and relationships described in the next section.

VERIFICATION & COMPLIANCE ASSESSMENT MODELS

A good verification/testing program is totally driven by requirements. Very simply, the overall flow of a verification program is as follows.

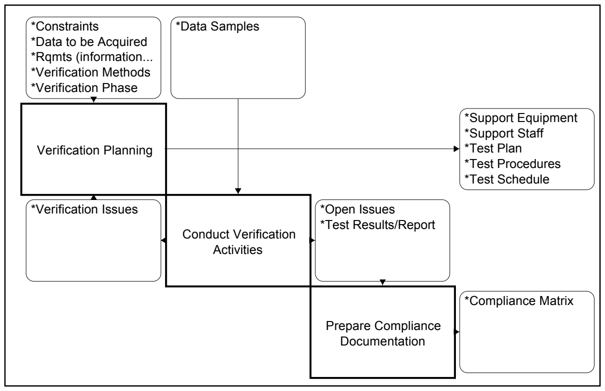


Figure 9 - Verification/Testing Plan Development

The availability of the executable information models speed development of the test plans, and improve the coverage due to the expanded visibility of the planner (i.e., the amount of reverse-engineering is minimized). Additionally, the ERA+Flows techniques are useful to lay out the required testing activities in a clear, understandable manor (see Figure 10). The CORE® automatic documentation generation capability is used to construct the proper test planning, test reports, and compliance matrix documentation.

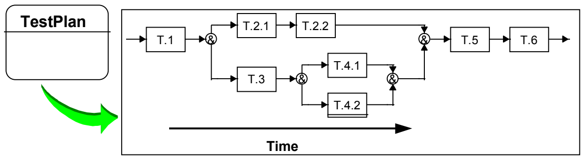


Figure 10 - Sample Test Plan Work-Flow

DOCUMENT GENERATION

The required reports/documents are generated from the engineering database, using the system engineering support tool's **Electronic Document Generation** capability. The tool generates standard system engineering, verification planning, and management visibility reports, as well as user-defined project specific reports/documents. The electronic documents consist of encoded text, tables, and graphics for easy interchange with other tools. Microsoft's Rich Text Format (RTF) standard is the interchange language supported by CORE®. Because the electronic documents (RTF files) are written in a standard interchange language, project personnel can share any textual, tabular, or graphical data stored in the ERA+Flows system design database, with organizations using different tools. The electronic document contents are fully editable, even the graphical system behavior flows. This eliminates, or at least greatly reduces, the need to manually build project review material and specifications independently from the engineering database where the majority of the data/information being reported already exists.

CONCLUSION

This paper has presented an automated process and supporting entity-relationship-attribute plus flow specification language to support the capture, management, and publishing of system design database knowledge. The SLM System Engineering Process, and supporting tool-set, is being used by many company's to implement a model-driven approach to system/process specification. The SLM Process, and tools, can also be used to prototype parts of INCOSE's Model Driven System Design (MDSD) process and methods discussed in reference 1.

REFERENCES

Baker, Loyd, Clemente, Paul, Cohen, Bob, Permenter, Larry, Purves, Byron, and Salmon, Pete; "Foundational Concepts For Model Driven System Design"; INCOSE Interest Group White Paper for 1996 International Symposium.

Long, David A., "RATM: Requirements Analyzer and Traceability Manager", Virginia Polytechnic Institute and State University, Master's Thesis, April 1992.

Baker, Loyd Jr., "Distributed Computing Design System: A Technical Overview, CDRL B003, TRW System Development Division, Huntsville, Alabama, July 1987.

ABOUT THE AUTHORS

Loyd Baker, Jr. is a Principal Engineer for Vitech Corporation, and serves as the co-chair of the INCOSE, Model Driven System Design, Interest Group.

James E. Long is the President of Vitech Corporation, the developer of the system engineering support tool, CORE®. Mr. Long is President-Elect of his local INCOSE Chapter.