GRAND SYSTEMS DEVELOPMENT TRAINING PROGRAM PRESENTATION

VERSION 14.0

**The union of sound system engineering,
domain engineering, functional management,
and program management
for the greater good of the enterprise
and customer base.**

VOLUME 122
THE MODEL, THE TEXTUAL AND GRAPHICAL RAS, AND THE SPECIFICATION – A LOGICAL AND EFFECTIVE PROGRESSION

Manual written by and presentation offered by
Jeffrey O. Grady
Owner JOG System Engineering
6015 Charae Street
San Diego, California 92122
(858) 458-0121
jeff@jogse.com
http://www.jogse.com

122
05-01-2014

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ACRONYMS

ACRONYM	MEANING	PAGE
DFD	Data Flow Diagram	25
DoDAF	Department of Defense Architecture Framework	1
FID	Function Identification	15
IDEF	Identification Definition	4
INCOSE	International Council On Systems Engineering	18
IRFNA	Inhibited Red Fuming Nitric Acid	9
MID	Modeling Identification	9
MoDAF	Ministry of Defence Architecture Framework	1
MSA	Modern Structured Analysis	1
MTBF	Mean Time Between Failures	7
PID	Product Identification	13
PSARE	Process for System Architecture and Requirements Engineering	1
RAS	Requirements Analysis Sheet	1
RID	Requirement Identification	13
SysML	System Modeling Language	1
UADF	Universal Architecture Description Framework	1
UML	Unified Modeling Language	1
UPDM	Unified Process for DoDAF MoDAF	1

The Model, the Textual and Graphical RAS, and the Specification, A Logical and Effective Progression

Jeffrey O. Grady
Owner JOG System Engineering
6015 Charae Street
San Diego, CA 92122
(858) 458-0121 jeff@jogse.com

1. Introduction

A system is a collection of product entities interrelated with each other and a prescribed environment through interface entities that collectively achieve a specific function. In the context of this paper these systems are developed by a program operated by a system development enterprise in accordance with a contract with a procuring customer. There are other possibilities but this is the context intended in this paper. A system and its subordinate entities are said to accomplish a well-defined functionality that should be fully explored and the requirements defining essential characteristics of the system captured in specifications for each entity in the system at some levels of indenture.

There are several ways that a development enterprise can determine an appropriate architecture and derive the content of the specifications it must prepare but this paper encourages that the architecture and all requirements for all entities be derived through one of three or four universal architecture description frameworks (UADF) each of which provides a comprehensive modeling capability that can be applied to all entities no matter how those entities are to be implemented in terms of hardware, software, or humans doing things. Three of these UADF are: (1) functional, (2) the combination of modern structured analysis (MSA) and process for system architecture and requirements engineering (PSARE), and (3) the combination of unified modeling language (UML) and system modeling language (SysML). It may be possible to form a fourth using unified process for DoDAF MoDAF (UPDM) but this paper does not pursue that alternative. Refer to the author's paper "Universal Architecture Description Framework" in Systems Engineering The Journal of The International Council On Systems Engineering, Volume 12 Number 2, Summer 2009 for details on UADF.

Figure 1 illustrates the way that effective models aid the development of a system concept and related requirements for inclusion in program specifications. The analyst stares at the problem space starting with the customer need, analyzes the problem space making sketches in a particular modeling approach that capture system functionality or behavior, the analyst derives performance requirements from the modeling artifacts, and allocates those requirements to entities in the evolving system product structure inserting new ones where nothing existing is adequate. The modeling is accomplished by the analyst manipulating images created with paper and pencil or computer screen and keyboard using hand-eye coordination in accordance with his or her mental activity associated with the modeling artifacts. Out of this effort comes a model of the system of increasingly expansive depth and breadth that tends to be complete and offer no unnecessary content. This can be a dynamic activity with lower tier modeling revealing better alternatives than selected in higher tier decisions previously arrived at.

The models available all employ a set of simple modeling artifacts that are intended to represent needed functionality, behavior, or physical attributes. Over time the problem space is translated into a series of simple sketches on the facets shown on Figure 1 that can be understood by those familiar with the model employed resulting in the disappearance of the problem space that can now be communicated effectively in the form of the sketches on these facets.

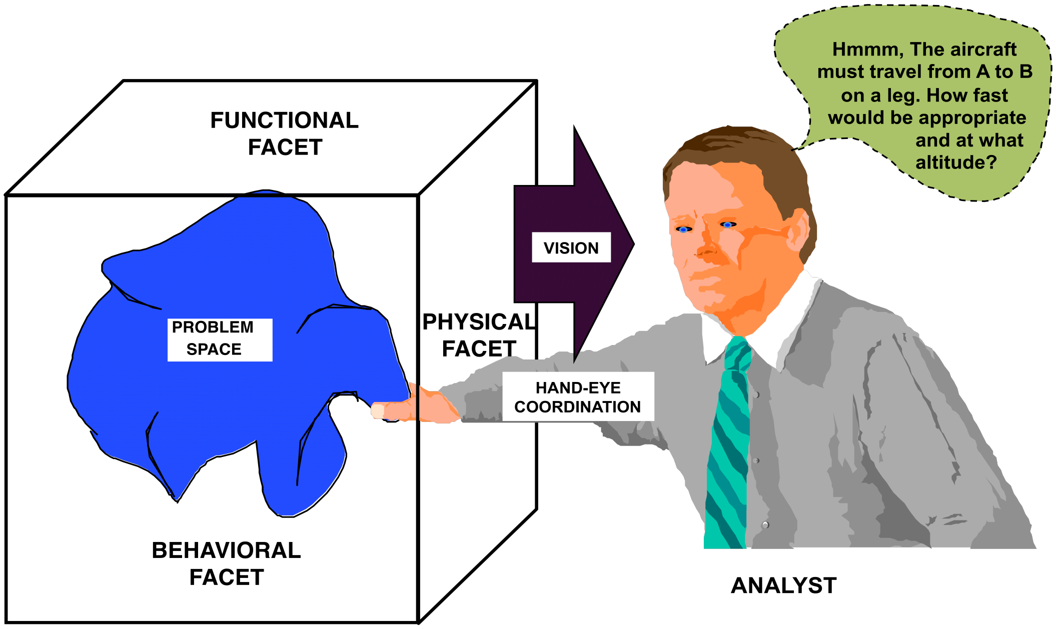

FUNCTIONAL FACET
PROBLEM SPACE
PHYSICAL FACET
BEHAVIORAL FACET
VISION
HAND-EYE COORDINATION
Hmmm, The aircraft must travel from A to B on a leg. How fast would be appropriate and at what altitude?
ANALYST

Figure 1 Modeling Fundamentals

In the preparation of a specification we should not only follow a well-described modeling approach, but leave behind a clear traceability record from the modeling through to the content of the specifications that is part of the formal configuration record. In the author's recollection he has never observed this to have been comprehensively done in the development of a system on a program. The rationale for following this pattern of behavior is that it is seldom that a system development process follows a uniformly perfect development process to a perfect result. When conflicts are identified it is helpful to possess a clear record of the path traveled such that errors in the path followed can be identified, corrected quickly, and adjustments made to restore program health. Also, respecting a formal traceability pattern of behavior encourages well-developed verification requirements that lead to effective and complete verification tasks the completion of which tends to provide good evidence of design compliance with the content of the specifications or early identification of problems that should be corrected.

Given that we accept that all requirements appearing in specifications be derived through modeling, it means that we must have a way to establish traceability between requirements and the modeling artifacts from which they were derived. This has seldom if ever been done well on a program. As noted above there are several effective modeling approaches, one of which our organization should excel in applying, and we should possess a traceability capability no matter which one we choose to apply. The author is certain that a pattern can be devised for the use of MSA-PSARE or UML-SysML but he has not published one to date. This paper describes the application of the encouraged pattern of behavior using the functional UADF. While it is true that software modeling was accomplished using this model in early software development work there are probably few enterprises that would select this UADF today because of an inability to gain software engineering acceptance. While MSA and UML were developed focused on the narrow needs of software development PSARE (or Hatley Pirbhai or HP as it was earlier known) and SysML were intended as system development models so either the MSA-PSARE or UML-SysML model can be used comprehensively by any enterprise with less disruption of a hardware dominated system engineering community, that may still be applying the functional model, than might be imagined.

This paper will cover some old ground with which the reader may be familiar but there are two new aspects of the presentation. One is the graphical construct used to communicate the relationships between the several elements of the modeling and requirements capture process. The author has used the basic elements of this construct before but even in his new book on requirements analysis to be published in 2013 by Elsevier he did not include one feature of it that woke him up in the dead of night to be captured when it was too late to submit it to the publisher. The weak element of the structure used in the book was the capture of environmental requirements and that is corrected in this paper by recognizing environmental entities as akin to product entities and the linkage between them in exactly the same fashion as interfaces with the product entities. The second is an explanation of how the human mind can apply a disciplined interface identification methodology evaluating all possible pairs of trios of modeling entities as a replacement for the intuitive method used by most experienced system engineers.

2. Modeling Overview

As noted above we will appeal to the functional model in this paper that consists of the functional flow diagram for gaining insight into needed performance requirements that when allocated to product entities begin the formation of the physical product entity structure for the system. There are three other models required, however, to complete the story. We need a means to identify and define all interface relationships between the entities. We also need a set of models to deal with the specialty engineering requirements appropriate for the system and its entities. Finally, we need a model for the system environment. The latter is one of the primary new elements of this paper. The paper will recognize environmental entities in a similar fashion to product entities that drive internal interfaces.

2.1 Functional Modeling

Figure 2 offers a view of system development popularized by Brian Mar and Bernard Morais in their writing and lectures. The functional model begins with a requirement often referred to as the system need that is derived from the ultimate function of a system represented by the peak of the functional pyramid in Figure 2 and it is allocated instantly to the system represented by the peak of the physical pyramid in Figure 2, the system. We continue to analyze functionality in layers of functional flow diagrams deriving requirements that are allocated to hierarchically expressed layers in the physical model. In the end, we have the physical product structure of the system (system entity model) defined as well as the performance requirements for those product and interface entities. Ideally, design concepts for product entities derived from functional modeling would be developed close behind the functional modeling with an intense interaction between the persons doing the modeling and those doing the concept development with system engineers and modeling and simulation people applying integration and optimization to the evolving picture. Proceeding in isolation in each pyramid is a process error.

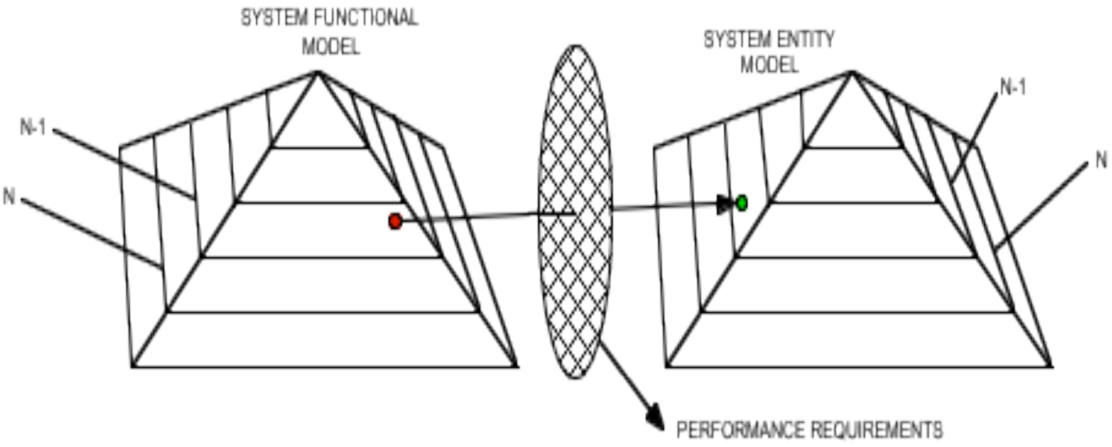


Figure 2 A Pyramidal View of Functional Analysis and Allocation

The principal modeling artifact employed in functional analysis is the functional flow diagram depicting blocks each one of which represents functionality that the system must be able to satisfy. These blocks are strung together in sequences by directed line segments that tell the order in which the functions have to be completed, some perhaps in parallel and others in a serial fashion. The analyst derives performance requirements from the functions that define what the system must be capable of doing and how well. These requirements are allocated to a product entity and the requirements captured in the specification for that entity.

Figure 3 shows an example of a functional flow diagram motivated by the General Dynamics Atlas Space Transport System. The upper stage referred to in this figure is essentially the same as the product entity depicted in Figure 4. Incidentally, the flow diagram provided in Figure 3 is not that different from the one appropriate for the complete Titan 4 Space Transport System only a part of which is shown in Figure 4.

There exist alternatives to the flow diagram including two axis sketches that also include supporting resource or data flow in the form of behavioral diagrams and enhanced functional flow block diagrams. IDEF-0 adds the flow of supporting resources and controlling influences to the flow diagram. Some analysts use a hierarchical functional diagram. When flow charts were employed in early software development diagramming they were drawn as vertical flow charts where the blocks represented needed computer software functionality from which requirements for computer code were derived. The activity diagram of UML and SysML is essentially a functional flow diagram.

Some analysts claim that the more complex models expose the analyst to a more complete context of the problem space and this may be true but they also can hide content from analyst’s appreciation because of the added complexity. Some analysts also prefer no more than some fixed number of blocks on any diagram with possibly more separate diagrams while others insist on fewer more comprehensive diagrams. There are advantages and disadvantages each way. Generally simple is good but the author tends to include too much content on each sheet.

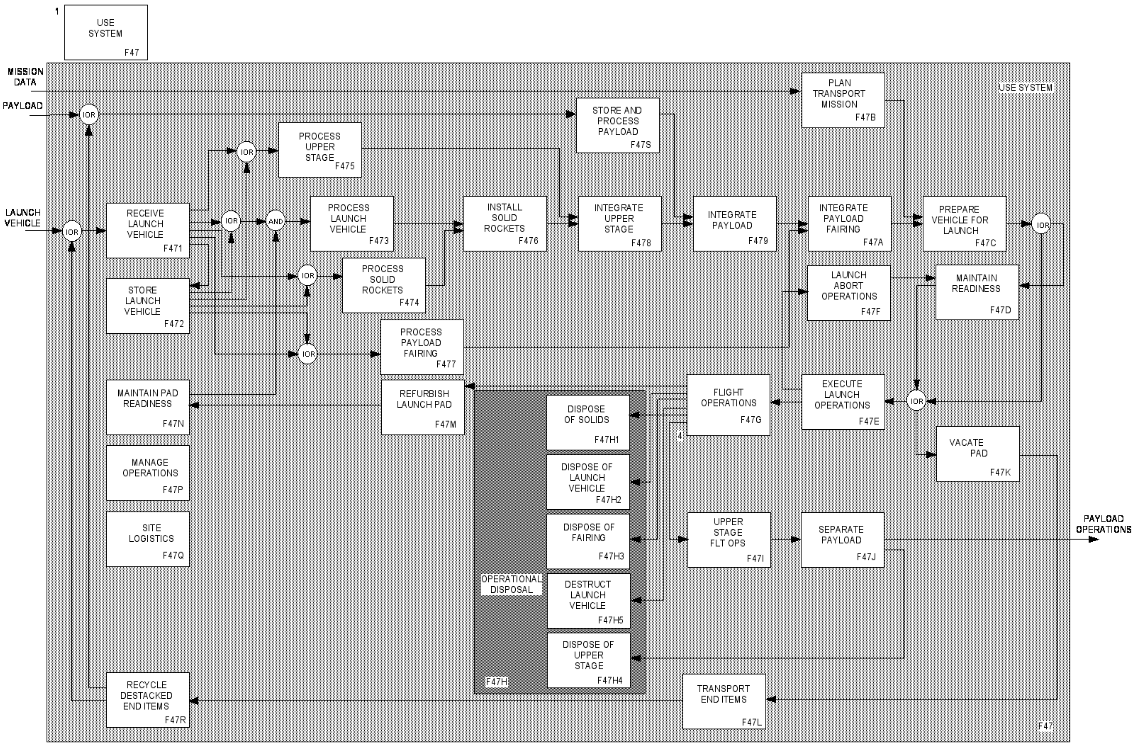

1
USE SYSTEM F47
MISSION DATA
PAYLOAD
LAUNCH VEHICLE
IOR
AND
RECEIVE LAUNCH VEHICLE F471
STORE LAUNCH VEHICLE F472
PROCESS LAUNCH VEHICLE F473
PROCESS SOLID ROCKETS F474
PROCESS UPPER STAGE F475
INSTALL SOLID ROCKETS F476
PROCESS PAYLOAD FAIRING F477
INTEGRATE UPPER STAGE F478
INTEGRATE PAYLOAD F479
INTEGRATE PAYLOAD FAIRING F47A
PLAN TRANSPORT MISSION F47B
PREPARE VEHICLE FOR LAUNCH F47C
MAINTAIN READINESS F47D
EXECUTE LAUNCH OPERATIONS F47E
LAUNCH ABORT OPERATIONS F47F
FLIGHT OPERATIONS F47G
4
OPERATIONAL DISPOSAL
F47H
DISPOSE OF SOLIDS F47H1
DISPOSE OF LAUNCH VEHICLE F47H2
DISPOSE OF FAIRING F47H3
DISPOSE OF UPPER STAGE F47H4
DESTRUCT LAUNCH VEHICLE F47H5
UPPER STAGE FLT OPS F47I
SEPARATE PAYLOAD F47J
VACATE PAD F47K
TRANSPORT END ITEMS F47L
REFURBISH LAUNCH PAD F47M
MAINTAIN PAD READINESS F47N
MANAGE OPERATIONS F47P
SITE LOGISTICS F47Q
RECYCLE DESTACKED END ITEMS F47R
STORE AND PROCESS PAYLOAD F47S
USE SYSTEM
F47
PAYLOAD OPERATIONS

Figure 3 Sample Functional Flow Diagram

2.2 Solution Space Models

The functional modeling is only intended to give us insight into performance requirements that tell what the system must do and how well it must do it. There are three other kinds of requirements we wish to capture in our specifications: (1) interfaces between product entities and their needed characteristics, (2) specialty engineering requirements, and (3) environmental requirements. The latter can be thought of as a special kind of interface requirement. The functional UADF includes models for all of these plus one to capture the intended physical structure of the system in the form of a product entity block diagram and another to define the physical process that the system will be employed in.

2.2.1 Product Entity

The product entity model is just a hierarchical block diagram beginning with a single block at the top for the system. Each level in the diagram has a block for each product entity at that level. Figure 4 shows a typical product entity diagram that happens to be for the Martin Marietta Titan 4 Space Transport Rocket System emphasizing the Centaur Upper Stage that at the time was built by General Dynamics Space Systems Division where the author was working at the time. At the time the author drew the original sketch for Figure 4 he believed that the modeling diagrams should be engineering drawings formally released but he has since concluded that they should either be captured in a special computer modeling application or published in appendices of a system architecture report. The content of the document in any case should be placed under configuration control possibly as an engineering drawing. The motive for the author changing his mind on this matter was that the complete model work product should be kept together.

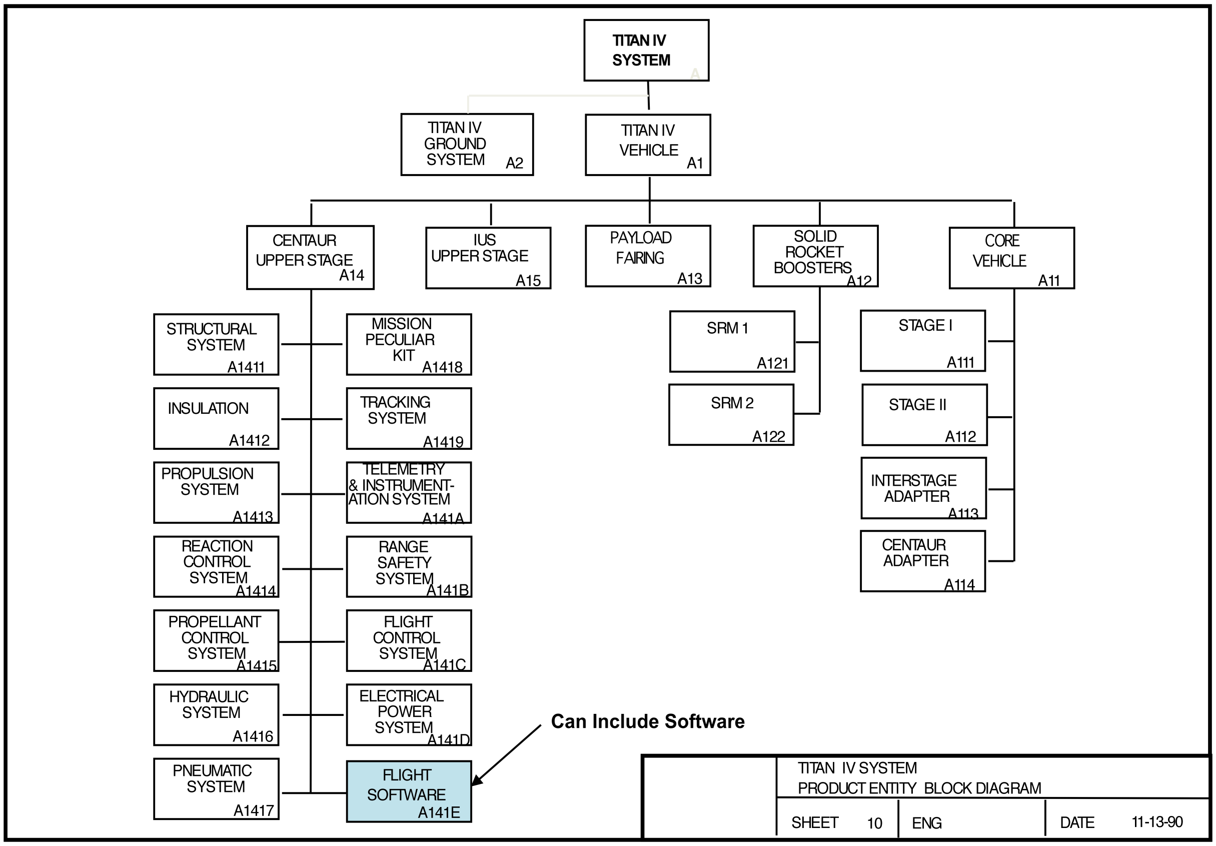


Figure 4 Typical Product Entity Block Diagram

2.2.2 Interface

The top end of the interface schematic block diagram model is illustrated in Figure 5. The I1 interfaces are internal to the system. Interfaces I2 are between environmental entities (modeling class Q of all kinds) and product entities (modeling class A). The I3 interfaces are between environmental entities on both terminals. Many system engineers would caution against recognizing any of the latter but the author claims that there are inter-environmental stresses that can be of value in the development of a system.

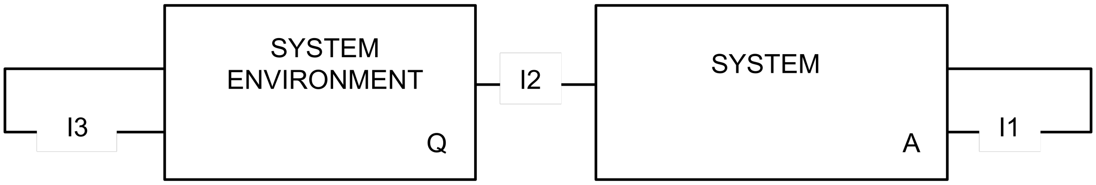
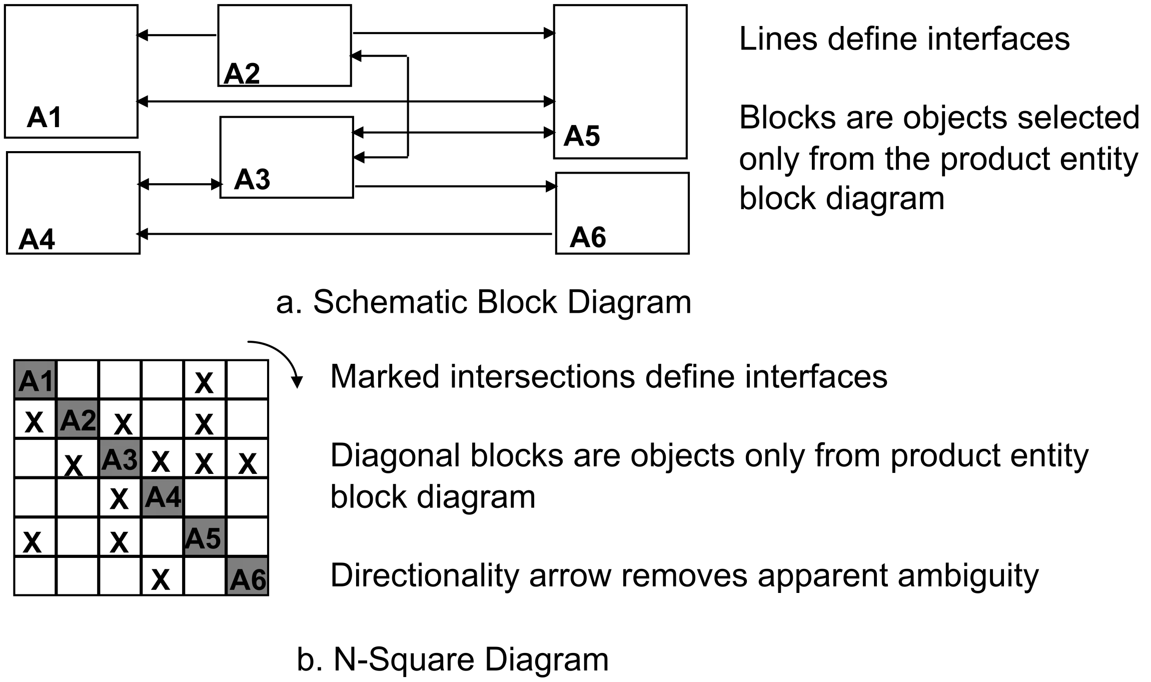


Figure 5 Top Level System Schematic Block Diagram

The functional UADF employs either a schematic block diagram or an n-square diagram to identify interfaces for which interface requirements are defined. In the schematic block diagram, shown in Figure 6a, product entities of interest are depicted by blocks from the product entity block diagram that are connected at the lower tiers by directed line segments to indicate a need for an interface. An n-square depiction of the interface is identical as is the one in Figure 6b relative to its schematic block diagram cousin where the product entities are marked on the diagonal. One should mark the intended directionality of the n-square diagram as the arrow in Figure 6b is intended to do. The reader can see from the n-square depiction that there should be an interface from A3 to A5 by the X marked in the appropriate intersection and confirm that this is also shown in the schematic block diagram. The problem with the diagrams noted in Figure 6 is that they do not help us to identify a need for an interface between two environmental entities or between a product entity and an environmental entity. These diagrams only provide an organized means for reporting interfaces. More on this a little later in this paper.


A1
A2
A3
A4
A5
A6
Lines define interfaces
Blocks are objects selected only from the product entity block diagram

a. Schematic Block Diagram

A1				X	
X	A2	X		X	
	X	A3	X	X	X
		X	A4		
X		X		A5	
			X		A6

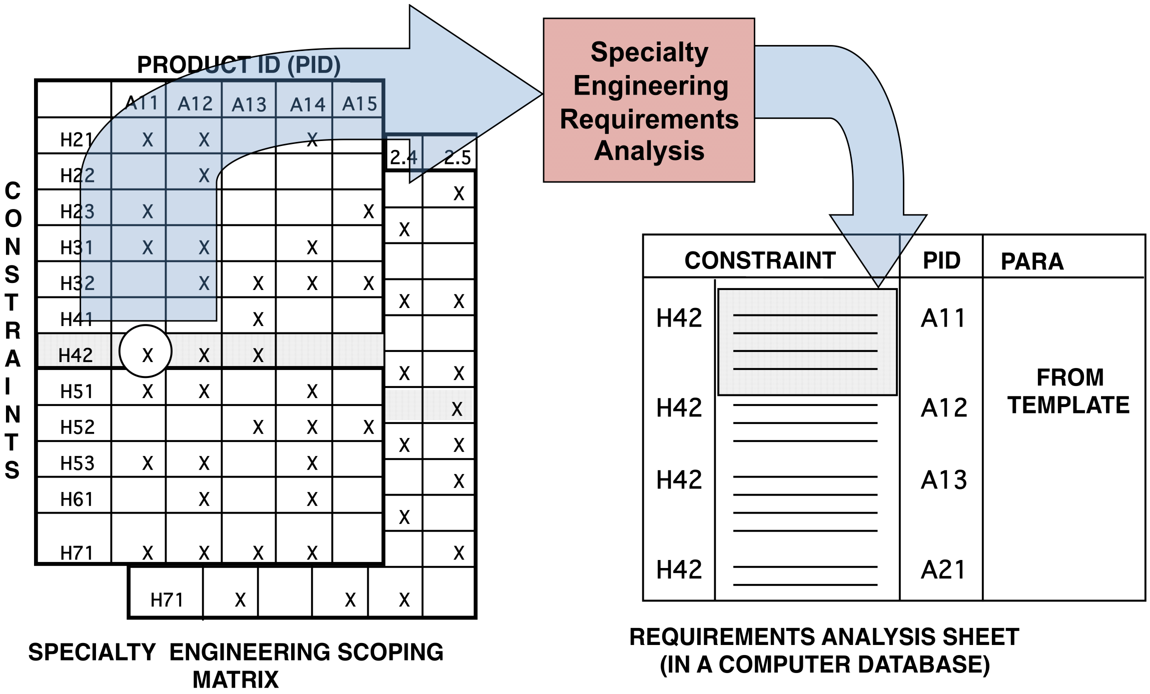
Marked intersections define interfaces
Diagonal blocks are objects only from product entity block diagram
Directionality arrow removes apparent ambiguity

b. N-Square Diagram

Figure 6 Interface Reporting Models

2.2.3 Specialty Engineering

There are many different specialty engineering domains such as reliability, maintainability, mass properties, and safety. Each of t hese domains has a particular modeling approach employed to derive requirements in their domain. For example, the reliability engineer creates a probabilistic reliability math model using either item failure rates, mean time between failure (MTBF) figures, or reliability percentages. The mass properties domain engineers use a weights statement. Figure 7 shows how the system engineer can encourage participation by the specialty engineering community using a specialty engineering scoping matrix to identify all of the product entities each discipline must craft requirements for. The author uses the lead letter H for specialty domain modeling IDs. The specialty engineer applies his or her model as directed by the specialty engineering scoping matrix to the product entities and derives appropriate requirements for inclusion in that specification. The resultant requirements are captured in the requirements analysis sheet implemented in a table, spreadsheet, or computer database. Each discipline should be required to configuration manage their model.


PRODUCT ID (PID)
CONSTRAINTS
2.4
2.5
H71
Specialty Engineering Requirements Analysis
CONSTRAINT
PID
PARA
H42
A11
H42
A12
H42
A13
H42
A21
FROM TEMPLATE
SPECIALTY ENGINEERING SCOPING MATRIX
REQUIREMENTS ANALYSIS SHEET (IN A COMPUTER DATABASE)

	A11	A12	A13	A14	A15
H21	X	X		X	
H22		X			
H23	X				X
H31	X	X		X	
H32		X	X	X	X
H41			X		
H42	X	X	X		
H51	X	X		X	
H52			X	X	X
H53	X	X		X	
H61		X		X	
H71	X	X	X	X	

Figure 7 Specialty Engineering Scoping Matrix

2.2.4 Environmental Requirements

The environment includes everything in the Universe less the system of interest. Most of the environment we can disregard, of course. We do have to decide the scope of the environmental space that must be considered. It is convenient to partition all environmental stresses into the several classes or entities shown in Figure 8. The natural environment consists of space, time, and natural stresses such as rainfall, Lunar or Martian dust, or atmospheric pressure as a function of the mission space of the system. There are standards for most localities with which we would ever have to deal so much of this work involves reading standards, tailoring them for the scope of our system, and adjusting the range of these variables for what we define as normal.

A standards model will satisfy most of our needs at the system level but a threat analysis will add good insights into any hostile intentions of adversaries. At the end item level it is helpful to build a three axis model inter-relating product entities, processes that will have to be applied to them, and environmental stresses applicable to those process steps that is referred to by some as an environmental use profile. It often happens that particular product entities have to be used in association with different sets of environmental stresses over the run of a mission. This model forces us to map product entities to the processes to which we have also mapped environmental stresses. From this model one can extract the union of all stresses applied in a mission for system end items.

At the component level one can partition an end item into zones of equal environmental stress and engage in a packaging study to determine where components will be installed. The components then inherit the zone stresses.

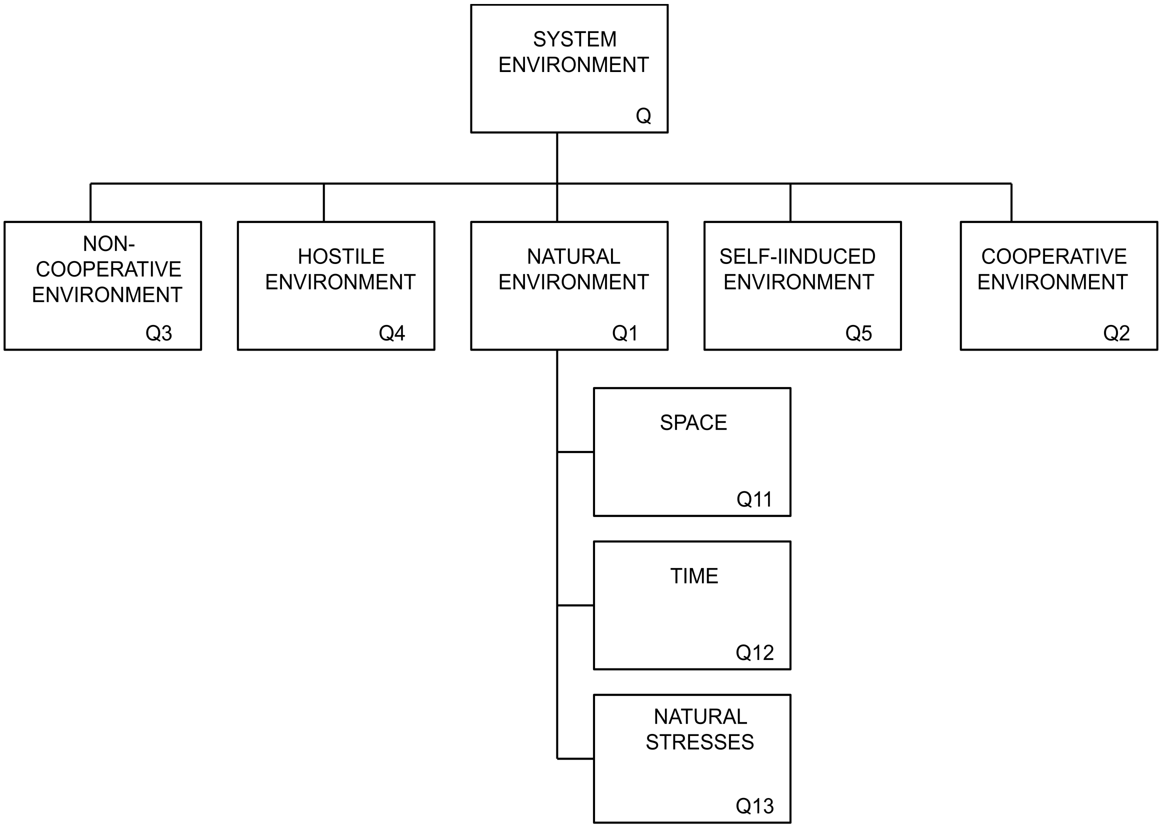


Figure 8 Environmental Classes

The cooperative environment consists of all of those systems intended to cooperate with the system being developed and these are most often developed as interfaces between a pair of systems. The non-cooperative interface consists of relationships between the system of interest and sources of stress applied to the system unintentionally. Hostile stresses come from other systems intent on damaging or reducing the effectiveness of our system. Self-induced stresses are initiated by the system being developed often from energy sources within the system such as a rocket engine causing tremendous acoustic vibration that effects everything in the vehicle while it is climbing through the atmosphere.

As noted, the cooperative environmental relationships are commonly developed as interfaces because there is another person or organization with which one can converse in a cooperative fashion even to the extent of developing a mutually respected interface specification. Where the author has gone astray in his modeling of environmental stresses in the past is that he has failed to treat the other environmental classes in exactly the same fashion. We will simply extend the system n-square diagram diagonal to include not only the cooperative system entities but all of the other environmental influences as well. This diagonal happens, of course to coincide with the product entity axis as it relates to the internal interfaces. So, we will extend this axis to include the environment which will coincide with the diagonal of an extended n-square diagram to include all external interfaces, not restricted to just the cooperative systems influences.

Figure 9 offers a schematic block diagram view of the relationship between the system of interest and the environment with all of the external interfaces assigned generic MID. We may choose to recognize the outerface lines with MID starting with "I3" or not but there are situations where it is very helpful. The figure also highlights the importance of the electromagnetic environment effects (E^3) classes that cut across the five environmental classes. Interestingly the E^3 military standard MIL-STD-464 is titled an interface standard. The author has become convinced that environmental requirements are a subset of interface requirements.

The internal interface requirements should be identified based on an evaluation of how needed functionality is associated with product entities through the allocation of performance requirements to those product entities. All external interface requirements can be derived from the functionality associated with the system environment through allocation of environmental requirements to environmental entities.

In Figure 9 we will apply functional analysis linked to product entities through performance requirements as the means to identify needed internal interfaces (I1) and functional analysis linked to environmental entities through environmental requirements to identify all external interface requirements (I2 and I3). We may choose to continue to refer to many of the external requirements as environmental requirements out of respect for the historical path we have followed to arrive at the present but if we choose we could rebuild the specification template to recognize only performance, interface, and specialty engineering requirements because all environmental requirements can be grouped with the external interface requirements.

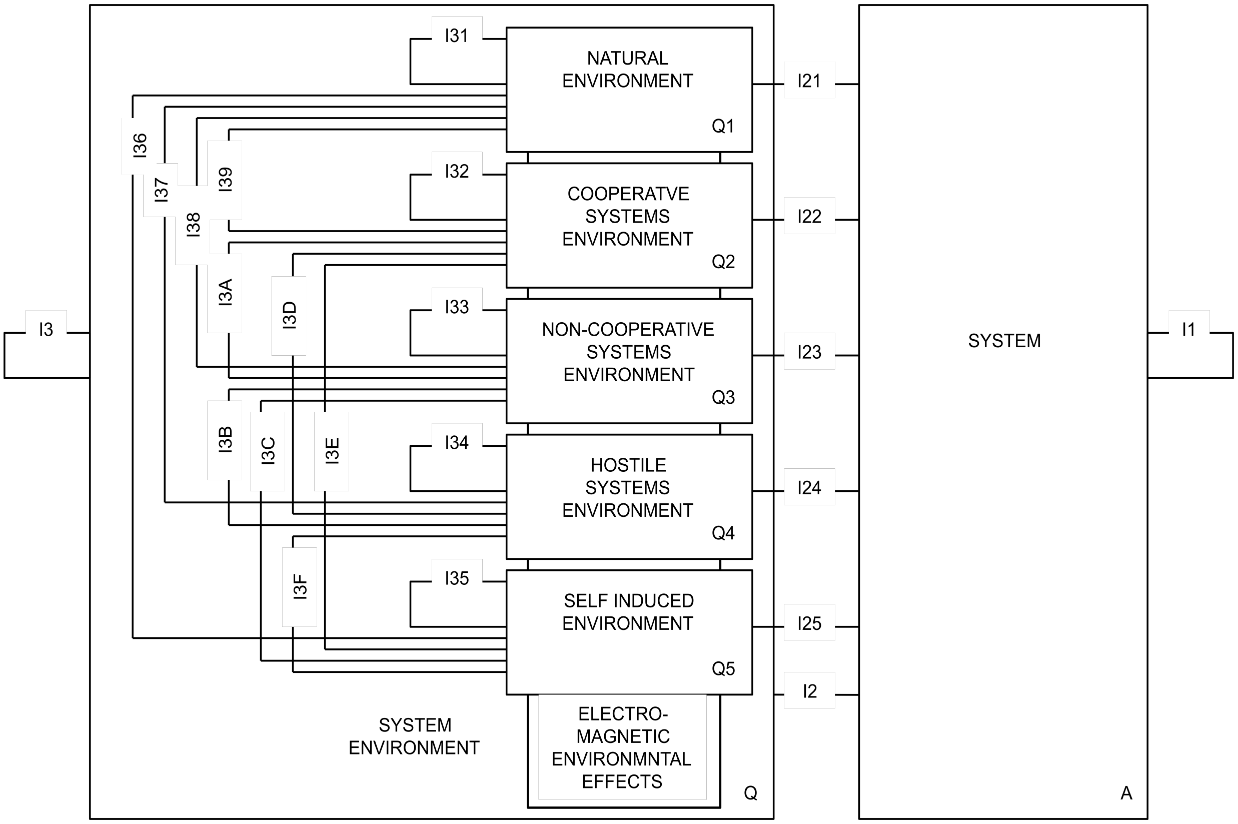


Figure 9 Environmental Relationships

2.2.5 Physical Process Model

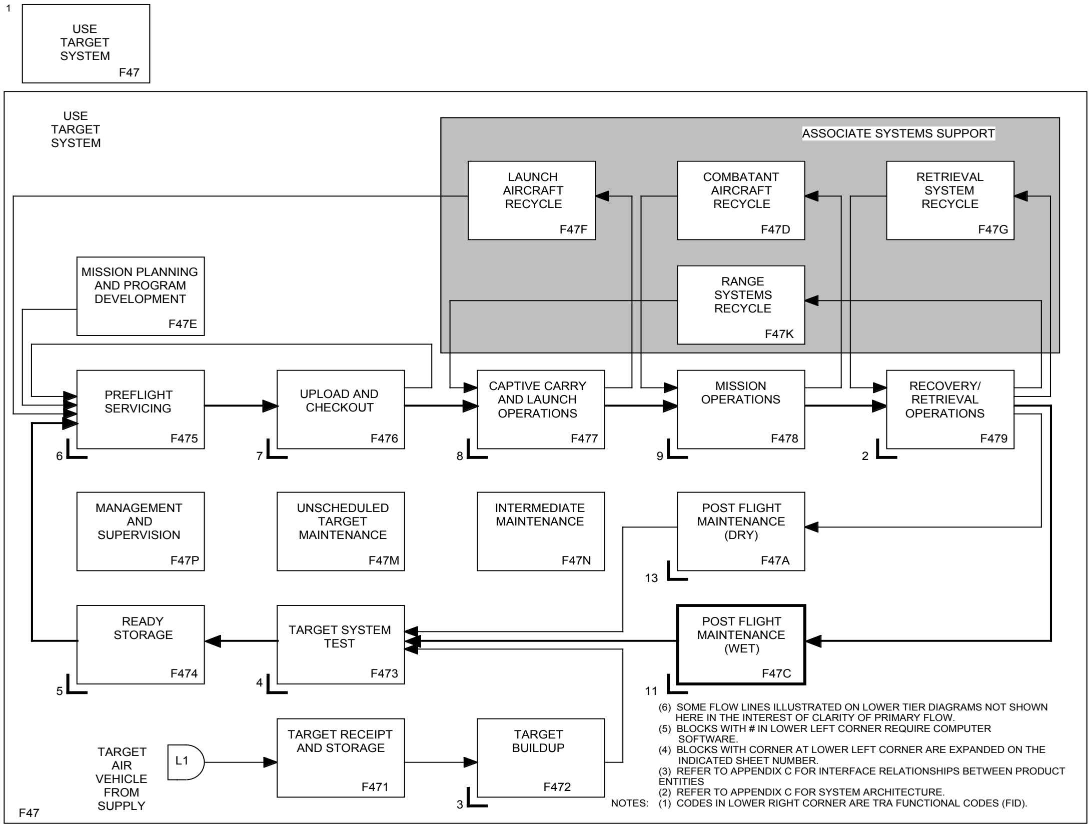
Figure 3 offered a functional flow diagram of a space transport system but if truth be told when a particular kind of system endures over a period of time the developing enterprise will often drift into the use of a process diagram which this figure has more in common with. The difference between functional flow diagram and a process flow diagram is in the meaning of the blocks. On a functional diagram the block represents a function or activity the system, or entity thereof, must complete or accomplish and at the time the diagram is created the developer commonly may have no idea what the system will consist of. The functional flow diagram is used as a means to determine what the system should consist of as described earlier. The blocks on a process flow diagram represent physical actions that the system, or entity thereof, must accomplish with full knowledge of what the system consist of. The physical process diagram is very useful in logistics analysis and does have an application in environmental requirements analysis.

For a heavily precedented system a process flow diagram could be employed rather than a functional flow diagram where as in the development of an unprecedented system there is no knowledge of what the system will consist of so we have to resort to the functional flow diagram and allocation as an organized means to determine the physical composition of the system. A locomotive company once hired the author to support them in their efforts to apply the systems approach in the development of a new diesel electric locomotive, something they had never done going back many years. They wished to treat the development as an unprecedented activity because their competition was beating their pants off on three requirements. The company actually employed a variation on PSARE without realizing it but with prior knowledge of the subsystems that would be necessary.

Figure 10 is a process flow diagram the author used in the development of the AQM91 Firebolt target drone. The blocks are identified with MID starting with the letter F but this is a physical process diagram. The authors employer had inherited the development of the system that the customer contracted with because of its displeasure of the prior developer. The design was adequate but needed to be more easily manufactured and maintained in the field as well as safer to operate. The propulsion system burned a rubber compound and inhibited red fuming nitric acid (IRFNA) in a ram jet engine to achieve Mach 4 at 100,000 feet. Therefore, it was possible to build a flow diagram reflecting what the previous contractor had tried to do so as to understand how the system could be improved. Several improvements resulted.

3. Modeling Artifact Identification

In order to be able to establish traceability between requirements and the modeling artifact it was derived from we have to be able to uniquely identify every modeling artifact from which requirements might be derived. The author uses what he calls a modeling ID (MID) for this purpose. His MID set for the functional UADF identifies artifacts by a alphanumeric string starting with a capitol letter corresponding to the artifact type as follows: (F) Function, (A) Product Entity, (I) Interface, (H) Specialty Engineering, (Q) Environmental, (P) Physical Process, and (R) Requirement. The reader can see MID examples of F, A, I, H, and Q in Figures 3, 4, 6, 7, and 8 respectively. The author would start a process flow diagram block MID with a letter P today but at the time he sketched the original Figure 10 he had not yet evolved a universal MID scheme. Individual modeling sketches should be relatively simple but it is not difficult to use up all 10 Arabic numerals on one sheet so the author uses a base 60 system employing the Arabic numerals, English alphabet capitol letters less “O” and lower case English alphabet characters less “l”. Each layer simply adds another place value.


1
USE TARGET SYSTEM
F47
USE TARGET SYSTEM
ASSOCIATE SYSTEMS SUPPORT
LAUNCH AIRCRAFT RECYCLE
F47F
COMBATANT AIRCRAFT RECYCLE
F47D
RETRIEVAL SYSTEM RECYCLE
F47G
RANGE SYSTEMS RECYCLE
F47K
MISSION PLANNING AND PROGRAM DEVELOPMENT
F47E
PREFLIGHT SERVICING
F475
6
UPLOAD AND CHECKOUT
F476
7
CAPTIVE CARRY AND LAUNCH OPERATIONS
F477
8
MISSION OPERATIONS
F478
9
RECOVERY/ RETRIEVAL OPERATIONS
F479
2
MANAGEMENT AND SUPERVISION
F47P
UNSCHEDULED TARGET MAINTENANCE
F47M
INTERMEDIATE MAINTENANCE
F47N
POST FLIGHT MAINTENANCE (DRY)
F47A
13
READY STORAGE
F474
5
TARGET SYSTEM TEST
F473
4
POST FLIGHT MAINTENANCE (WET)
F47C
11
TARGET AIR VEHICLE FROM SUPPLY
L1
TARGET RECEIPT AND STORAGE
F471
TARGET BUILDUP
F472
3
F47

(6) SOME FLOW LINES ILLUSTRATED ON LOWER TIER DIAGRAMS NOT SHOWN HERE IN THE INTEREST OF CLARITY OF PRIMARY FLOW.
(5) BLOCKS WITH # IN LOWER LEFT CORNER REQUIRE COMPUTER SOFTWARE.
(4) BLOCKS WITH CORNER AT LOWER LEFT CORNER ARE EXPANDED ON THE INDICATED SHEET NUMBER.
(3) REFER TO APPENDIX C FOR INTERFACE RELATIONSHIPS BETWEEN PRODUCT ENTITIES
(2) REFER TO APPENDIX C FOR SYSTEM ARCHITECTURE.
NOTES: (1) CODES IN LOWER RIGHT CORNER ARE TRA FUNCTIONAL CODES (FID).

Figure 10 Process Flow Diagram

There is not a lot of history at this point on the use of this technique resulting in any kind of convention so the enterprise adopting this approach should simply establish its own preferred assignments. Table 1 shows a partial list from the author’s new “System Requirements Analysis” book due from Elsevier in late 2013 derived from the author’s consulting company JOG System Engineering. The paragraph number (PARA) included is the specification template paragraph number where related requirements should appear. The department (DEPT) column gives the department number for the functional department from which personnel should be acquired by a program to do the related work. The PREFERRED MODEL column tells the functional UADF model those people should apply on a program. In the book noted, this table is three pages long including some MD related to MSA-PSARE and UML-SysML UADF so the table has been truncated here to only give a few examples.

Table 1 JOGSE Universal Modeling ID List Sample

MID	MEANING	PARA	DEPT	PREFERRED MODEL
A	Product Entity	3.1	331	Product Entity Block Diagram
F	Functionality	3.1	331	Functional Flow Diagramming
H	Specialty Engineering Domain	3.4	331	Specialty Engineering Scoping Matrix and Specialty Models
H1	Engineering Domains	3.4.1	3	-
H11	Aerodynamics	3.4.1.1	321	Modeling and Simulation
H12	Thermodynamics	3.4.1.2	322	Thermodynamic Analysis
H13	Structural Integrity	3.5.1.3	323	Modeling and Simulation
H14	Structural Statics	3.5.1.4	323	Modeling and Simulation
H15	Structural Dynamics	3.5.1.5	323	Modeling and Simulation
H2	Logistics Domains	3.4.2	4	-
I	Interface	3.3	331	Coordinated N-Square Diagram
I1	Internal Interface	3.3.1	331	Coordinated N-Square Diagram
I2	External Interface	3.3.2	331	Coordinated N-Square Diagram
I3	Outside Interface	3.3.3	331	Coordinated N-Square Diagram
P	Process		331	Process Flow Diagram
Q	Environment	3.5	331	-
Q1	Natural Environment	3.5.1	331	Standards
Q11	Space	3.5.1.1	331	Mission Analysis and Packaging
Q12	Time	3.5.1.2	331	Time Lines
Q13	Natural Stresses	3.5.1.3	331	Standards
Q2	Cooperative Environment	3.3.2	331	N-Square Diagram
Q3	Non-Cooperative Environment	3.5.2	331	Threat Analysis
Q4	Hostile Environment	3.5.3	331	Threat Analysis
Q5	Self-Induced Environment	3.5.4	331	-

4. Specification Templates

A system development enterprise should possess a set of specification standards that is supportive of their customer base preferences possibly tailored to reflect their own preferred practices. The author’s preference is MIL-STD-961E tailored to coordinate the Section 3 structure with the model selected by the enterprise as suggested in Table 2. If the enterprise were applying the functional model the Section 3 structure would follow the pattern shown below:

3. REQUIREMENTS
3.1 Modeling
3.2 Performance Requirements
3.3 Interface Requirements
3.3.1 Internal Interface Requirements
3.3.2 External Interface Requirements (Cooperative Systems Environmental)
3.3 Outside Interface Requirements
3.4 Specialty Engineering Requirements
3.5 Environmental Requirements
3.5.1 Natural Environmental Requirements
3.5.2 Non-Cooperative Environmental Requirements
3.5.3 Hostile Systems Environmental Requirements
3.5.4 Self-Induced Environmental Requirements

The author confesses to a great temptation to include all environmental requirements under paragraphs 3.3.2 and 3.3.3 covering interfaces of the type I2 and I3 in Figure 5 respectively. Then under each of these paragraphs would be paragraphs 3.3.X.1 through 3.3.X.5 corresponding to natural, cooperative systems, non-cooperative systems, hostile systems, and self-induced respectively. The paragraph 3.5 would be deleted because all of the environmental requirements would be covered as interfaces. The resistance to surrendering to this temptation is that there is a long tradition in recognizing the special nature of environmental relationships. We have no obligation to the past however to fail to see the reality that environmental relationships can be usefully represented as interfaces.

5. Requirements Analysis Sheet (RAS)

The RAS ties together the modeling sources, specification template structure, and requirements content of the specifications. It can be captured in a paper and pencil table, computer spreadsheet, or computer database constructed with records and fields. Each record should be a unique requirement for a particular entity. Every requirement in the system should be in this RAS. If you capture the RAS in a database then it is possible to not only retain the content under configuration control but actually publish the specifications. To print a specification one simply orders the database content by paragraph number and sets the filter for printing only content corresponding to a particular product entity. The specification will come out of the printer in paragraph number order. One can purchase database systems that can support a wide range of capabilities beyond these simple ones. The author actually prefers a primitive structure where the database captures the essential information and computer code combines it to form English sentences but Table 2 is based on capture of requirements in complete sentences.

A primitive requirement statement replaces the TEXT field in Table 2 with a series of four fields: (1) Attribute that tells what must be controlled, (2) Relationship that tells how the attribute is related to the numerical value, (3) Value that gives a numerical figure, and (4) Units that tells what units the value is measured in. For example the statement “Weight ≤ 134 Pounds” is an example of a primitive statement. Yes, these statements may take on a more complex form than this simple view including a range of values or a tolerance but those extensions can be easily handled. There are also cases where the requirements are qualitatively stated but this too can be dealt with in the primitive structure. The attributes in these statements come from modeling work. The values and relationships come from the application of good engineering skill to the problem and the values of related requirements in the parent item specification.

The advantage of a primitive statement is that the value can be stated in a numerical field and a database system within which the requirements are captured can manipulate these numerical values for many useful purposes such as searching for opportunities to deal with margins. Writing the code to string the primitive statement into a normal English sentence is not difficult. One could even argue that there is no real need to do so unless you are wedded to traditional specification structures. Figure 11 offers an example of a single record in the RAS. There are many other fields that the RAS could benefit from and the reader can think of some of them easily. Vertical traceability and verification traceability are a couple of fields that come to mind.

Table 2 RAS Record Structure

FIELD	FIELD TITLE	DATA TYPE
PARA	Paragraph Number	Decimal delimited numerical string
TITLE	Paragraph Title	Text with leading letters capitalized
FID	Modeling ID	MID string of base 60 characters
PID	Product Entity ID	MID string of base 60 characters beginning with A
RID	Requirements ID	System unique string of characters identifying the requirement
TEXT	Requirements Text	Text providing the requirement statement in complete sentences

In that there are several MID in the tabular RAS that must appear in every record it is necessary to refer to those several MID uniquely. We can use the terms FID, PID, and RID for functional, product, and requirement ID respectively for this purpose.

FIELD	REPRESENTATIVE FIELD CONTENT EXAMPLE
PARA	3.2.5.1
TITLE	Aircraft Speed
FID	F4723
PID	A1
RID	R7Y5j6S
TEXT	The aircraft shall be capable of flying at Mach 1.3 in level flight at 23,000 feet above mean sea level in a no wind condition.

Figure 11 Example of RAS Content

6. The RAS In Graphical Form

In that the human mind if so effective in understanding concepts presented in graphical form, an attempt has been made to state the RAS in the form of sketches. Once again, this example is employing the functional UADF. We will use a three-dimensional structure where points, lines, and planes have particular significance relative to the tabular, text-based RAS and the modeling work that drives RAS content. The reader will observe four planes in the graphical RAS corresponding to the four kinds of requirements must be derived. Generally points on these planes represent specific requirements and the axes correspond to particular modeling artifacts like functions and product entities. The author has built a similar structure for the MSA-PSARE UADF and will apply it to the UML-SysML UADF shortly.

6.1 The Performance Requirements Plane

The first component of the graphically expressed RAS is formed by a map between functions and product entities where intersection points on the resultant plane represent performance requirements. Figure 12 shows this construct. The reader will note that function “F” is used as the basis for deriving the Need Statement that is instantly allocated to the system assigned MID “A”. Two other functions have been identified with performance requirements derived that are allocated to product entities.

All of the performance requirements derived from functions and allocated to product entity A43 will appear in the item A43 specification with paragraph numbers beginning with 3.2. Note in Figure 11 that paragraph number 3.2.5.1 was assigned. The order in which they appear is up to the person doing the work but an organization in a particular product field could make some rules for this. The performance requirements are identified with a MID known as a Requirement ID that is assigned randomly but uniquely that is used for linking requirements for traceability purposes. Paragraphs numbers can change in a specification but Requirements IDs should remain stable under changes to the requirements and their paragraph numbers. If these Requirement IDs (RID) are unique for all requirements across a whole system, one might question the availability of enough unique RIDs for all of the specifications required on a program. In this case we are using a six-place (leading R same for all requirements) base 60 ID resulting in 60^6 RIDs or 46,656,000,000 requirements. It is only Sections 3 and 4 that would need RID assigned so if we exceed this number for any system we have been too busy writing requirements.

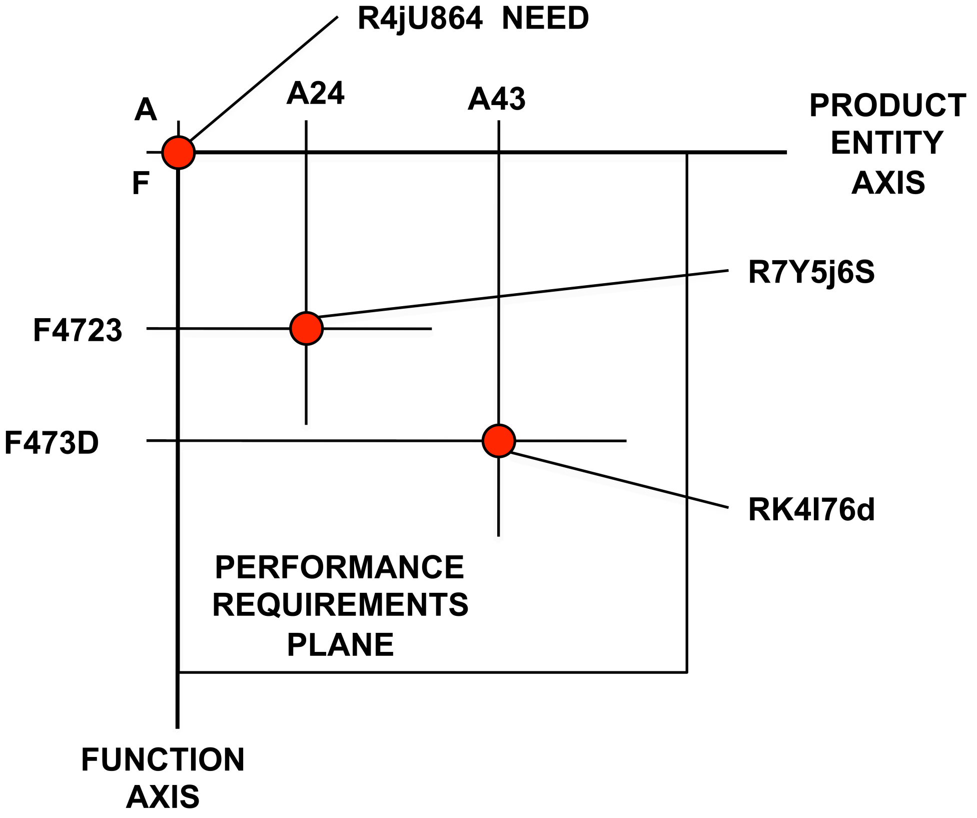


Figure 12 Performance Requirements Plane

6.2 Interface and Environmental Coverage

6.2.1 The Interface Plane

An n-square diagram can be used to depict all interface requirements but we have to orient it correctly for maximum benefit. This diagram can be started with two blocks on the diagonal one for the system represented by product entity “A” and the other by environmental ID “Q” for the whole environment as shown in Figure 13a. The expansion of the n-square diagram for the internal interface for the system might evolve as shown in Figure 13b and further showing the internal and external subsystem interfaces in Figure 13c.

It is important to recognize that the interface identification will expand from top down as the product system structure expands under continued functional analysis. We identify needed interfaces between the indicated product entities by marking the squares off the diagonal. An arrow at one corner indicates the intended directionality meaning that all marked squares above and to the right of the diagonal mean one direct of source and destination and those marked below and to the left of the diagonal the other. Squares unmarked mean that there is no corresponding interface identified as of that date. Each marked square represents one or more interface requirements each of which would bare a requirement ID linked in the RAS where the corresponding requirement statement would be included.

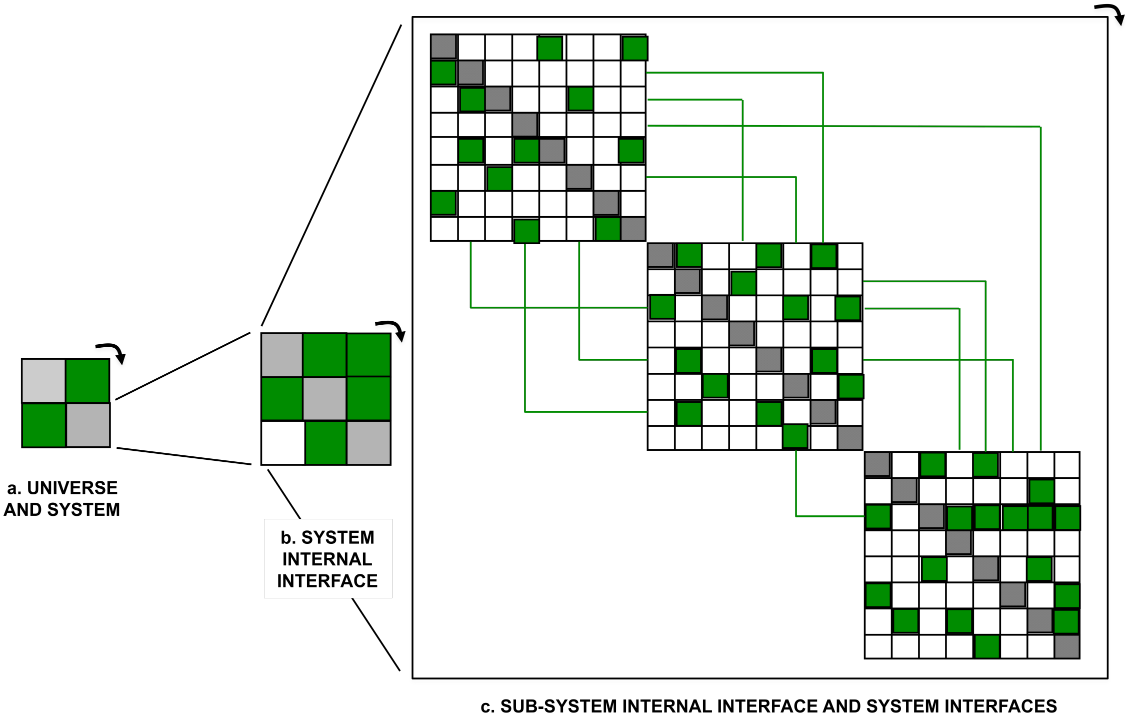

a. UNIVERSE AND SYSTEM
b. SYSTEM INTERNAL INTERFACE
c. SUB-SYSTEM INTERNAL INTERFACE AND SYSTEM INTERFACES

Figure 13 Progressive Interface Decomposition

As a system development program continues to define the architecture of the system using a disciplined top-down modeling approach it may be realized at a lower tier that an interface is needed that was not accounted for at a higher tier and the interface baseline may have to be rippled to account for the change. While not a desirable outcome, the synthesis work at any time may also discover as the work proceeds that a great opportunity was missed in earlier work and the baseline should be changed to reflect having taken advantage of that opportunity. It cannot be overstated how important it is that the architecture and requirements engineering work be coordinated with the trailing system synthesis work developing design concepts thought to be compliant with requirements. Having a clear modeling baseline under configuration control is every bit as important as having the design drawings under configuration control.

The reader should note that the diagonal of an n-square diagram can be marked with the product structure because from an interface perspective it represents the internal interfaces of the product entities identified on the diagonal. As the systems development process continues we can continue to expand the n-square diagram as suggested on Figure 13 where what is one diagonal block for a product entity on Figure 13b becomes expanded to reveal 8 subordinate entities on Figure13c representing the immediately subordinate entities for that subsystem.

Ideally, there would be a lot of internal interfaces at lower tiers with relatively few interfaces between higher tier entities. Figure 13b is characterized by all but one possible interface being needed at that level. This is often a signal that there will be a lot of difficulty in developing subordinate entities because of the interface intensity. The reader can see from Figure 13 how the system engineer can use an n-square diagram to help determine where the developing system will most need his or her help. Combine this view with the number of different disciplines needed on the related teams that drives the number of knowledge domains involved and a fairly clear picture of needed program system engineer loading can be observed because system engineers should live at the product and knowledge boundaries.

It is true that identification of needed interfaces in the functional model is a little strained but they are all pre-determined by the way that we have associated functionality with the product entities. It is part of the early system synthesis work to build the interface needs as the product entity structure is defined as a result of functional development, performance requirements derivation and allocation to product entities thus continuing the advance the expansion of the lower tier of product structure. At one INCOSE Symposium the author interviewed several long time system engineers he respected about how they identified interfaces using the functional model and they all replied that it was obvious that particular interfaces were needed and none of them had any special method. It was just obvious. It probably is for any one with a great deal of experience in a particular product field but the secret is that one must remain attuned to the way functionality is allocated to product entities to be able to appreciate those obvious intuitions.

Figure 14 shows how interface requirements appear on the interface plane coordinated with the product entities that appear as the terminals for those interfaces. These requirements are entered into the RAS linked to their RID. The plane has been rotated 45 degree to the left in preparation for the next step. In Figure 14, note the directionality arrow that means in this case that A24 is the source terminal for the interface and A43 is the receiving terminal. If the directionality were the opposite of this the interface would have been identified in the corresponding square on the other side of the diagonal. If the interface were bi-directional (shown on a schematic block diagram with an arrow on both ends of the line) it would be shown in both product entity pair of interface intersecting blocks.

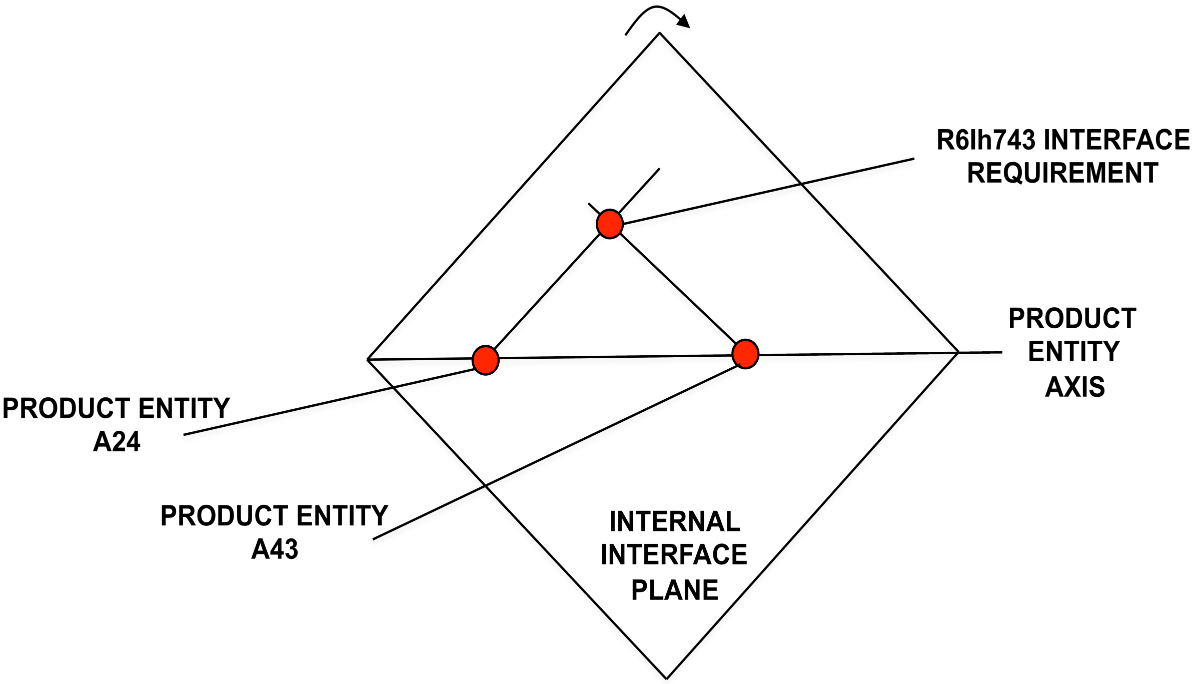


Figure 14 Rotated Internal Interface Plane

The bi-directional case raises another issue and that is whether we should recognize it as a single interface RID or two, one for each direction. It is true that the character of the interface could be significantly different in the two directions. For example, DC power could be supplied from A24 to A43 in the case of Figure 14 and a signal could be superimposed on this same line coming from A43 back to A24. This latter interface situation is not, of course, shown on Figure 15 so does not exist in this example. The author is inclined to assign different RID for the two directions in a bi-directional interface case even where the character of the interface is identical such as two way flow of data of the same structure on a bus.

6.2.2 Orientation of the Product Entity Internal Interface Plane

It was noted earlier that it was important to recognize the orientation of the n-square plane and the importance in this orientation is in recognizing that the n-square diagonal happens to coordinate with the performance requirements plane product entity axis. If we lay the diagonal onto the product entity axis as in Figure 15 then the off-diagonal squares represent the possible interfaces between these product entities.

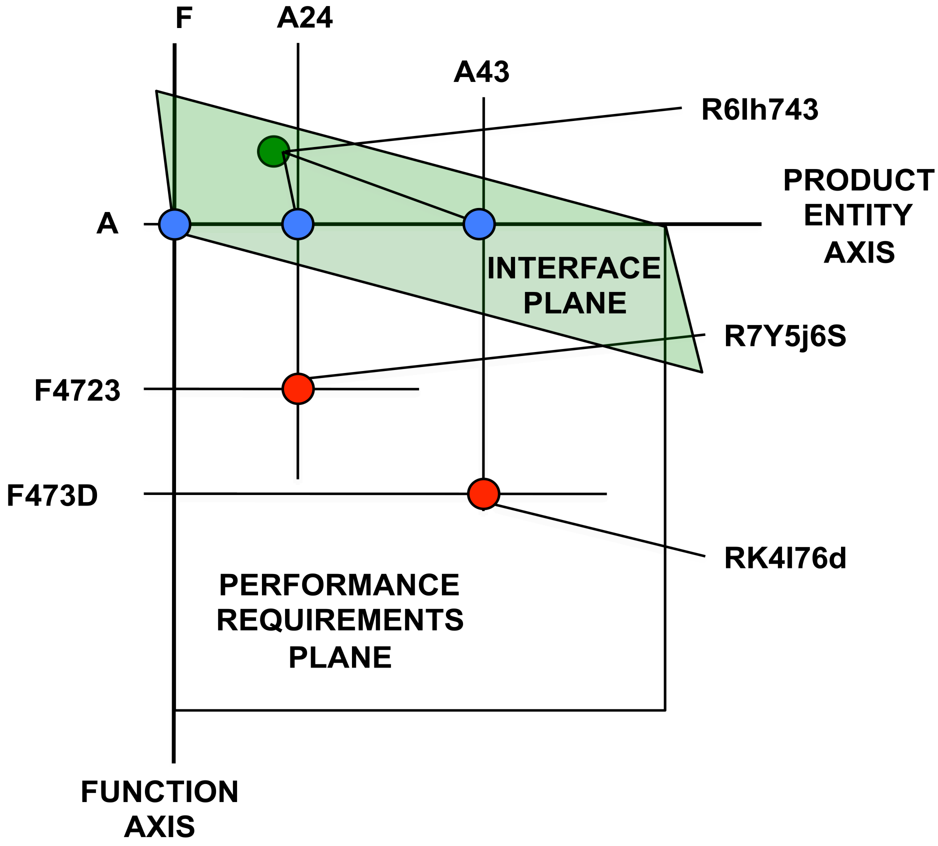


Figure 15 Internal Interface Plane Orientation

6.2.3 Extension of the Internal Interface Plane to Cover the Environment

The author has included some very tortured attempts to illustrate the environmental requirements in a graphical image of the RAS in the past and apologizes for inflicting them on any readers unfortunate enough to have been exposed to them. It turns out to be very simple to include them in the graphical RAS and this is the motive for offering this paper in the first place. The attentive reader will have recognized that so far we have not dealt with external interfaces, only internal ones. We aligned the diagonal of the interface plane with the product entity axis of the function allocation plane. That plane you will note has also not recognized entities in the environment of the system so far. So let us extend the product entity axis of that plane to recognize environmental entities to which functionality may also be allocated as shown in Figure 16. We can also partition that axis extension to recognize the environmental classes, or entities, shown in Figure 8. Now we may extend the interface plane to cover both internal and external (environmental) entities. The internal interface plane only identifies interfaces I1 shown on Figure 5. We may now identify interfaces I2 between system entities and environmental entities. We may also identify interfaces I3 between environmental entities on both terminals if we choose. We may also identify those interfaces with RID in our RAS and assign paragraph numbers in the specifications for the interface requirements whether they be derived from functionality allocated via performance requirements or environmental requirements.

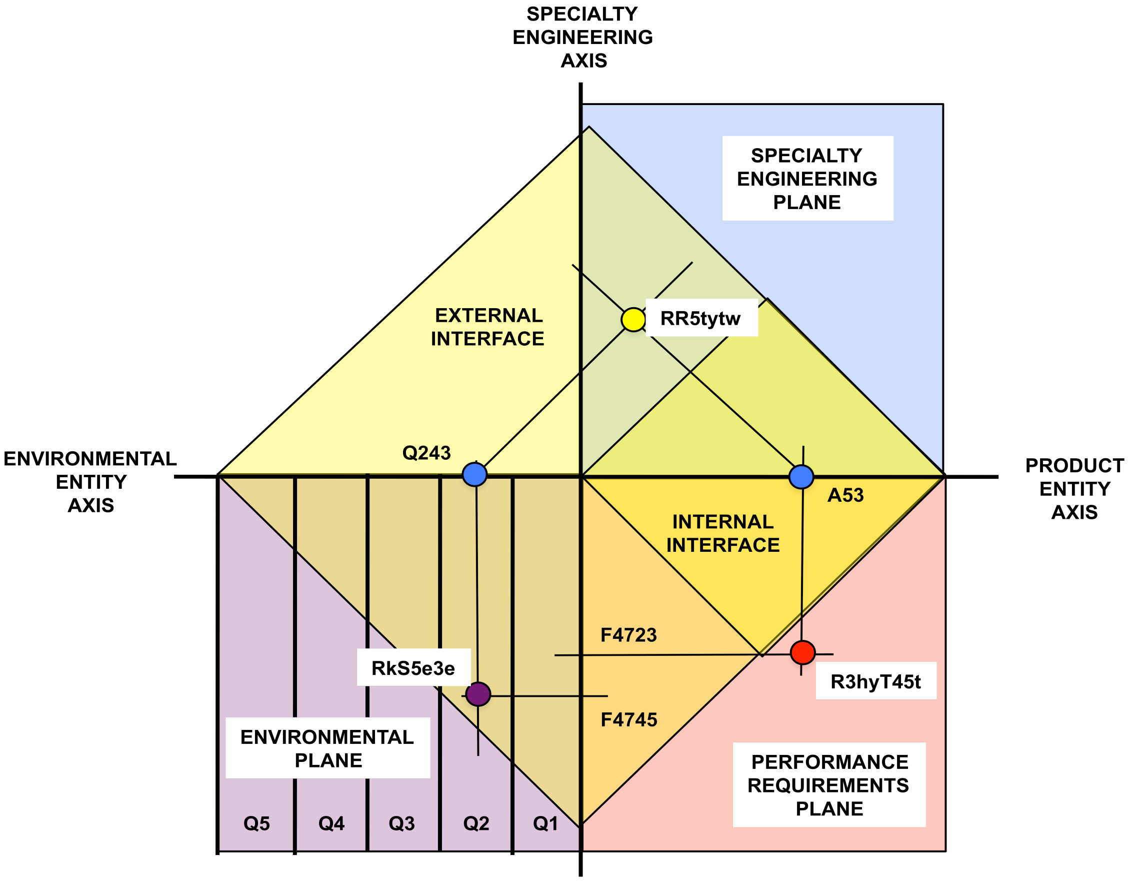


Figure 16 Extended Interface Plane

Note that while the rotated extended interface plane does allow us to visually appreciate the total interface issue in the development of any system, it does not solve the problem of how to explain how the engineer determines in an organized fashion what interfaces will be needed based on knowledge of how performance requirements derived from functionality were allocated to product and environmental entities. We will take up that issue shortly after dealing with the specialty engineering plane.

In Figure 16 we have done one more thing that is not very conventional. We have established a relationship between environmental entities and system functionality. Certainly, you would agree that there should be a relationship between system functionality and the entities comprising the cooperative environment, Q2. The thinking process for establishing these relationships is essentially the same as that for product entities but there may be many cases where we recognize the need for a cooperative entity first by recognizing an interface needed to support a product entity interface. Clearly, there should be a relationship between system functionality and the environment more generally. Because we intend to fly in the atmosphere recognized in system functionality there are many Earth atmosphere entities that have to be coordinated with product entities including lift provided by air motion relative to wing surface characteristics. It should be noted that our two-dimensional paper has forced the author to lay the interface plane in the same plane as the other planes in Figure 16 and that it is intended that the interface plane would share the horizontal axes with the other planes but be 90 degree rotated from that plane.

6.3 Specialty Engineering Plane

Our evolving graphical RAS in now complete with one exception. We have not yet dealt with specialty engineering requirements but this can done by adding a specialty engineering plane like that shown in Figure 16. One axis includes all of the specialty engineering domains we identified for our system in the specialty engineering scoping matrix in Figure 7. The other axis is the product entity axis that coincides with the other applications of this axis in Figures 12, 15, and 16. Each marked intersection on the plane, illustrated in isolation in Figure 17, corresponds to a specialty engineering requirement identified with a RID and associated with a RAS entry that flows into an item (A43 in this case) specification. All of the specialty engineering domains allocated to the line A43 represent the set of specialty engineering requirements that will appear in the specification for that item under Paragraph 3.4. The specialty engineer for the domain in each case will apply the domain model to derive the requirement value and enter it in the RAS linked to the appropriate entity.

6.4 Complete Graphical RAS

Figure 18 shows all of the parts of the graphical RAS. The figure is split into three views to portray each plane properly. The interface plane diagonal is, of course, coincident with the product entity axis extended to include the environmental entity axis but is 90 degrees rotated as indicated in the three-view drawing.

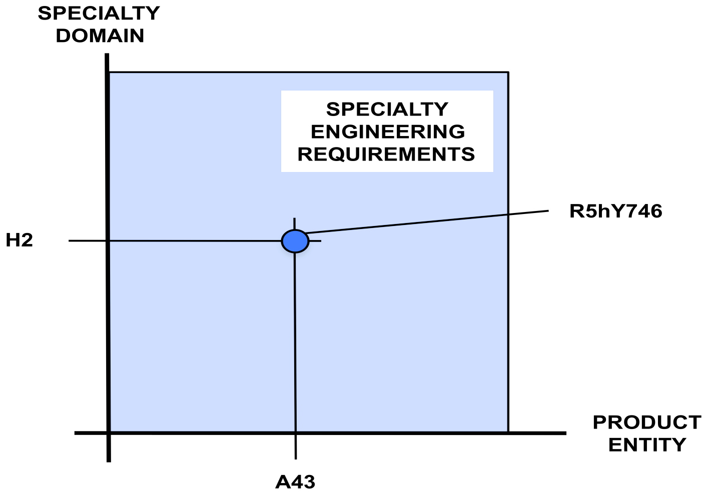


Figure 17 Specialty Engineering Plane

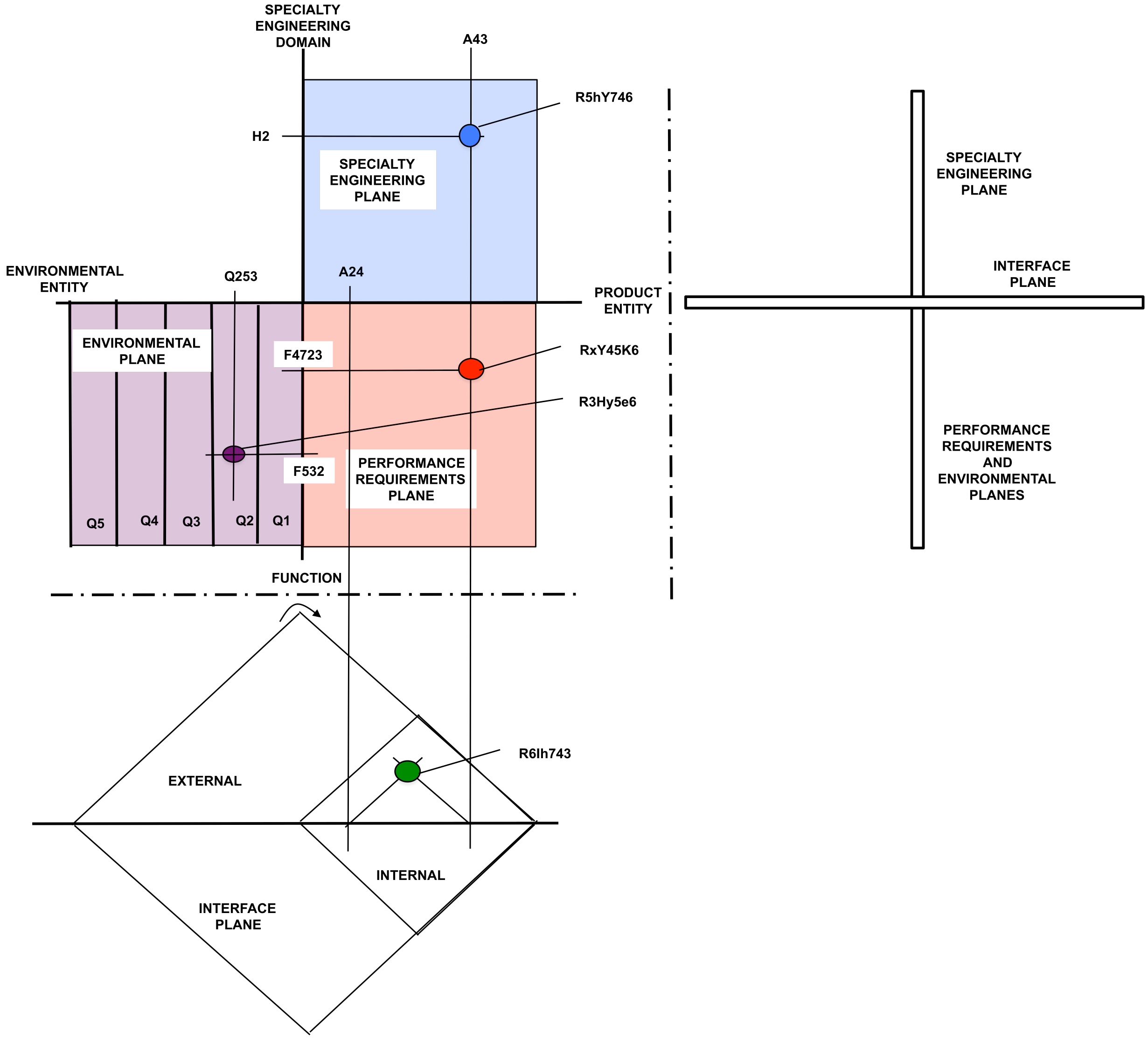


Figure 18 Complete Graphical RAS Structure

7. Disciplined Interface Identification Algorithm

Figure 15 shows a pair of the function, performance requirement, and product entity strings where it has been determined by whatever means that the pair has established a possible demand for an interface between product entities A24 and A43 with A24 as the source to which an interface requirement has been identified and assigned requirements ID (RID) R6Ih743. The reality is that an n-square diagram is not an analytical model rather a reporting medium. The question remains, how did the mind of the system engineer actually reach a conclusion that an interface is necessary in this situation. Every function-performance requirements-product entity trio does not necessarily demand identification of an interface but every needed interface will be pre-determined by a pair of these trios.

It is difficult to create a view showing that we must evaluate every pair of these trios for each set of functions that we derive performance requirements from and allocate to product entities. Figure 15 is a three-dimensional construct shown in two and if we place enough information on the figure to show even three or four trios to display more than a single interface it will be very hard for the reader to understand the message displayed. So, the reader will have to try to master the idea that it may require the system engineer to evaluate 1000 trio pairs to identify 100 interfaces.

So, we might ask, how many combinations would I expect to have to evaluate to make this technique effective? Well, the number of intersections on a functional plane is simply the number of functions at that level times the number of product entities identified at that time. If we were dealing with say 8 functions and 8 product entities, the number of intersections in which performance requirements could be placed would be 64. Let us assume that the functional analysis has resulted in the 20 performance requirements having been derived and allocated to product entities. The number of combinations of 20 elements (n) taken 2 at a time (k) is n!/k!(n-k)!. In this case the number of combinations would be 20!/2!(20-2)! = 2.4329x10^{18}/2(6.4024x10^{15} = 190. So, to fully implement the suggested analysis for interface identification in this case it would involve evaluation of 190 pairs of trios for interface need. There may be a conclusion on some of these cases that there is no need to mark a need for in interface on the n-square diagram and in other cases an off-diagonal intersection will have to be marked. Thus the n-square diagram becomes a means of reporting needed interfaces rather than a medium in which the interface analysis is conducted under this method.

It is through this process of evaluating pairs of trios that we can mark the intersections of the n-square diagram reporting the need for indicated interfaces finally answering the question of how the human mind determines that an interface is needed between two entities. The very experienced system engineer commonly need not depend on an organized and exhaustive study such as that suggested here but the mind of a good system engineer is subconsciously recognizing the functional relationships between the evolving product entities. Every trio pair does not an interface demand but every interface is pre-determined by a trio pair.

In each case one has to determine which product entity is the source and which the destination, of course, and the media in which the interface will be completed (electrical, mechanical, etc). Yet it is still not as simple as described so far. It happens as noted that there is parallel synthesis work going on at this time to come up with design concepts that have a chance of complying with the requirements identified by that time. The anticipated concept implementation will have a lot to do with selecting the medium of the interface, of course. In the process of applying this intense evaluation of possibilities the system engineer might in addition to identifying needed interfaces supportive of the current and building design concept gain insight into some interface possibilities that would be supportive of growth capabilities some already conceived and others that no one has thought of.

Clearly this algorithm can be extended to the external interface requirements aligned with environmental entities and requirements for both class I2 and I3 interfaces. Most system engineers, and even more so program managers worried about the number of man hours required to accomplish his work, would conclude that we don't have to apply this exhaustive algorithm to be successful so long as we are managing a well conceived modeling approach coordinated with a well done early synthesis activity. Unfortunately this combination is less often achieved than we would like.

8. Verification Requirements

The discussion in prior paragraphs covers all of the content of Section 3 of every program specification. Each of these specifications should, of course, include requirements in Section 4 for the verification requirements crafted for each product requirement appearing in Section 3. Since the graphical RAS we have created is a three-dimensional artifact it is not so easy to simply add another plane for verification requirements and link it to the product requirements. We can, however, support this need with a simple traceability table showing the Section 3 requirements in one column and the Section 4 requirements in another. This can actually be satisfied in the text-based RAS as noted. There is a stronger story that is required here, that will not be developed in this paper, to link the verification requirements to the content of the verification task plans, procedures, and reports prepared for each of those verification tasks.

9. Summary, Prescription, and Closing

This paper has offered a comprehensive story of the flow of work performed by a system engineer early in a program when the architecture of the system is being established and requirements derived from the related modeling entities and included in specifications for the system entities. Figure 19 is an attempt to graphically present this story tying the modeling work more closely with the RAS than the author had previously succeeding in doing. We model the problem space and derive all requirements that will appear in program specifications from those modeling artifacts. The model used in the paper for Figure 19 has been the functional one. If employing the MSA-PSARE model we would employ a data flow diagram (DFD) in which the bubbles represent functionality and we overlay that functionality with super bubbles to allocate the functionality to product entities. The bubbles are joined by the directed line segments that immediately identify needed interfaces between the super bubbles. PSARE permits the bubbles to represent hardware as well as software functionality and interfaces other than just data.

In any case performance requirements are derived from the bubbles and their relationships, each uniquely identified, and included in the RAS linked to the bubbles or functions from which they were derived. The RAS includes the uniquely identified modeling artifacts from which we derive performance requirements. These performance requirements are allocated to specific product entities about which specifications are prepared and these specifications are prepared in accordance with a template giving us the paragraph numbers for the requirement derived from the modeling work. The same general process could be employed if applying the UML-SysML UADF using a combination of activity, sequence, and state diagrams among others to gain insight into performance requirements. These other two UADF include product entity and interface modeling but are weak in specialty engineering and environmental modeling but those modeling components can be applied from the functional UADF.

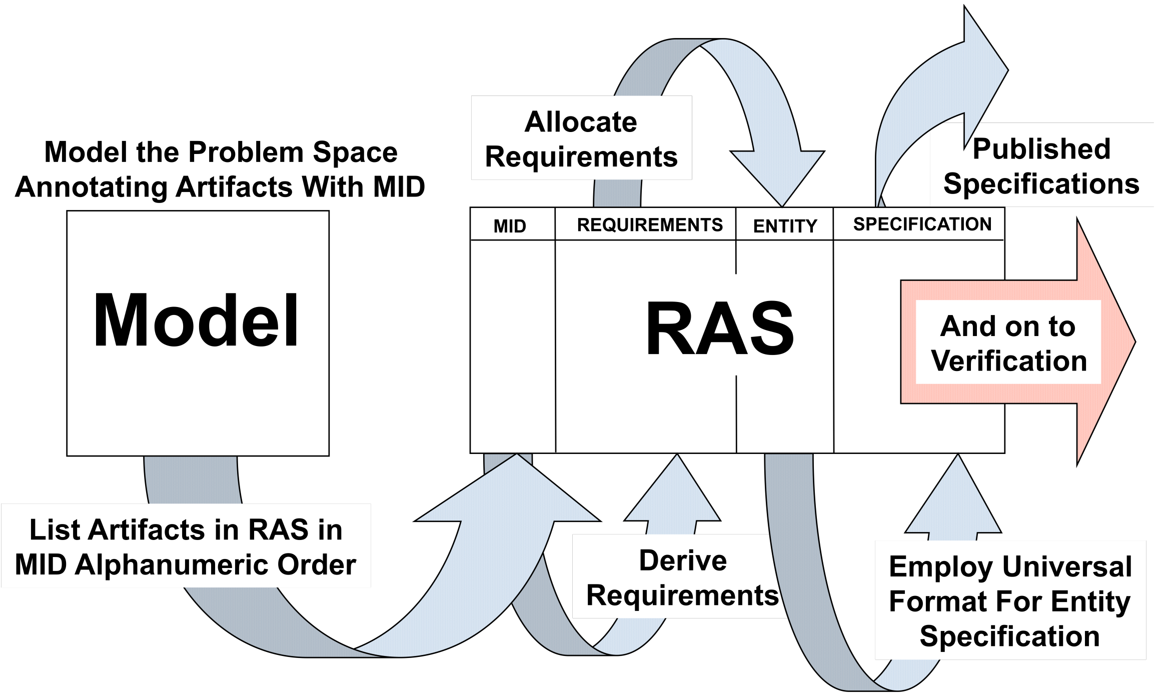


Figure 19 The Resultant Progression

The author believes that this pattern could also be applied if employing UPDM expressed in UML-SysML modeling artifacts. It is not believed that all 52 DoDAF diagrams would be necessary to support the requirements engineering work that may be required to comply with customer needs for architecture development and communications. If the government maintains an interest in DoDAF the ultimate modeling approach may become UPDM expressed in UML-SysML with a subset employed to comprehensively develop the content of the specifications.

This paper has focused on requirements work and given the companion synthesis work insufficient coverage. The reality is that the view offered in Figure 2 is critically important. As the requirements work for one layer of system architecture is completed the program must take action on the next layer of synthesis work. Whether this involves immediate product plane work following the completion of that functional plane or some delay like two layers rather than one is a matter of the art of system engineering. This is also the area that must be further explored to fully develop how the mind of the system engineer may act in an orderly fashion to identify the need for a particular interface using the functional UADF. The reader has hopefully observed the ease with which needed physical interfaces are determined from MSA-PSARE modeling compared to the opaque window through which the system engineer must gaze when applying the functional model. This comparison is a little unfair to the functional model perhaps in that it is easy to identify physical interfaces in MSA-PSARE after the DFD has been overlaid with super bubbles but the problem remains essentially the same as in the functional model if one steps back to how the analyst identifies the need for the directed line segments in the DFD in the first place.

The author's experience over many years has been that few programs do a good job in implementing the work discussed in this paper. So, we might ask, how would an organization improve it's ability to do this work well. The author's prescription follows:

1. Adopt a UADF and insist that all persons doing architecture development and requirements analysis use it on every program.

2. Adopt a way of uniquely identifying all modeling artifacts from which requirements may be derived.

3. Adopt a means by which personnel may capture modeling and specification content such that they may be configuration managed. There are not any computer tools known to the author that could capture all of the modeling and documentation features covered in this paper but one could build a simple text-oriented database linked to hand drawn or computer application graphical modeling artifacts.

4. Adopt a means for personnel to accomplish modeling work and retention of masters in the formal system baseline documentation that can be configuration managed.

5. Adopt a set of specification templates and to the extent possible develop a companion set of data item descriptions that tell how to translate a template into a program specification. The specification templates required include the following (items b, c, d, and e may require hardware and software versions but should apply the same UADF in deriving content):

    a. System Specification
    b. Item Performance Specification
    c. Item Detail Specification
    d. Interface Performance Specification
    e. Interface Detail Specification
    f. Part Specification
    g. Material Specification
    h. Process Specification

6. Establish a policy such as Table 1 suggests that clearly assigns responsibility for all specification content to personnel from specific functional departments on all programs. Departments identified will be responsible for preparing their department members for performing the related work on programs.

7. Prepare a written document telling how this work is to be done on programs.

8. Train all personnel who have a role in this work in the appropriate parts of it assigned to their functional department.

9. Establish a quality assurance means that will assure that the work is accomplished in accordance with the prepared instructions.

Managers in industry who are disappointed by program performance over the years should be especially inclined to look into ways of improving architecture and requirements performance because it is the beginning of success or failure on a program. So often managers find themselves late in a program with few desirable courses of action because of choices made very early in the program that by the time serious program problems have appeared they cannot be easily traced to poor requirements work performance. Often this is realized during the time verification work is going on and there simply are not enough money, time, or customer good will remaining even if a suitable corrective action were known. Previously in Paragraph 8 the author noted that a stronger story was needed to extend the traceability record beyond the discussion in this paper. The author's paper titled "Affordable Requirements Verification" appearing in the July 2013 (Volume 16 Issue 2) INCOSE Insight magazine covers a final solution to the problems often occurring on programs during verification.

JOG SYSTEM ENGINEERING
GRAND SYSTEMS DEVELOPMENT
TRAINING PROGRAM
INTRODUCTORY PRESENTATION

VOLUME 122
THE MODEL, THE TEXTUAL AND GRAPHICAL RAS, AND THE SPECIFICATION - A LOGICAL AND EFFECTIVE PROGRESSION
STUDENT MANUAL

EXHIBIT A
PRESENTATION MATERIALS

JOG SYSTEM ENGINEERING
GRAND SYSTEMS DEVELOPMENT
TRAINING PROGRAM INTRODUCTORY PRESENTATION

THE MODEL, THE TEXTUAL AND GRAPHICAL RAS, AND THE SPECIFICATION – A LOGICAL AND EFFECTIVE PROGRESSION

Presented By
Jeffrey O. Grady
(858) 458-0121
jeff@jogse.com

Who Is Jeff Grady?

CURRENT POSITION

- 1993 – PRESENT
  - Owner, JOG System Engineering
  - System Engineering Assessment, Consulting, and Education Firm

PRIOR EXPERIENCE

- 1954 - 1964 U.S. Marine Corps
- 1964 - 1965 General Precision, Librascope Division
  - Customer Training Instructor, SUBROC and ASROC ASW Computing Systems
- 1965 - 1982 Teledyne Ryan Aeronautical
  - Field Engineer, AQM-34 Series Special Purpose Aircraft Systems
  - Project Engineer, System Engineer on Unmanned Aircraft Systems
- 1982 - 1984 General Dynamics Convair Division
  - System Engineer, Cruise Missile, Advanced Cruise Missile
- 1984 - 1993 General Dynamics Space Systems Division
  - Functional Engineering Manager Systems Development Department

FORMAL EDUCATION

SDSU BA Math, UCSD Systems Engineering Certificate,
USC MS Systems Management With Information Systems Certificate

INCOSE Founder, Fellow, ESEP, and First Elected Secretary

AUTHOR System Requirements Analysis (3), System Integration, System Validation and Verification, System Verification, System Engineering Planning and Enterprise Identity, System Engineering Deployment, System Synthesis, System Management

The Principal Presentation References

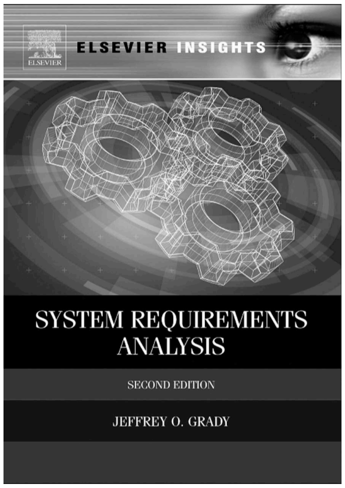
“System Requirements Analysis, 2nd Edition”, Jeffrey O. Grady, Elsevier Academic Press, 2014

"The Model, the Textual and Graphical RAS, and the Specification – A Logical and Effective Progression", Jeffrey O. Grady, paper not yet published, 2013

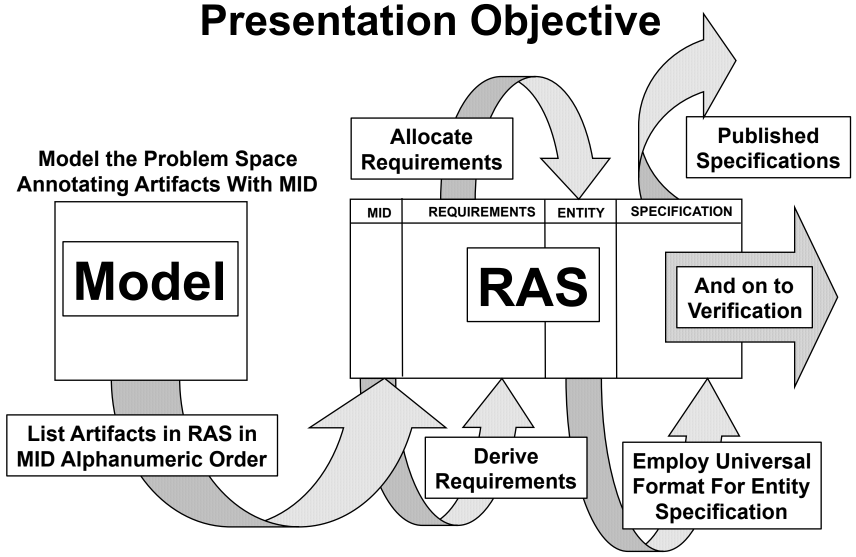
"Universal Architecture Description Framework (UADF)", Jeffrey O. Grady, Systems Engineering, The Journal of The International Council On Systems Engineering, Volume 12 Number 2, Summer 2009 (Best Paper 2009)

"Affordable Requirements Verification", Jeffrey O. Grady, INCOSE Insight, July 2013 (Volume 16, Issue 2)

New System Requirements Analysis Book in E-Book Format



Presentation Objective



Model the Problem Space Annotating Artifacts With MID

Model

List Artifacts in RAS in MID Alphanumeric Order

Allocate Requirements

MID

REQUIREMENTS

ENTITY

SPECIFICATION

RAS

Derive Requirements

Published Specifications

And on to Verification

Employ Universal Format For Entity Specification

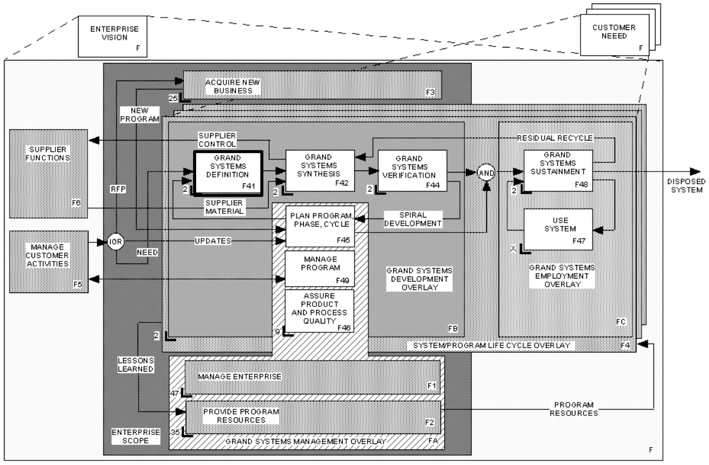
What Is a System?

- Collection of product entities intended to achieve a specific function
- Immersed in an environment
- Product and environmental entities inter-related through interfaces
- Product and interface entities clearly defined in a set of specifications where all of the content has been derived though application of a model to the problem space

Systems Development

- Define the problem to be solved in a set of product and interface entity specifications
- Solve the problem through synthesis in a three-step process
  - Design
  - Procurement
  - Manufacturing
- Determine extent to which entities and the system comply with the content of the specifications through verification
- Manage the program well throughout its development period

Enterprise Common Process View of System Life Cycle



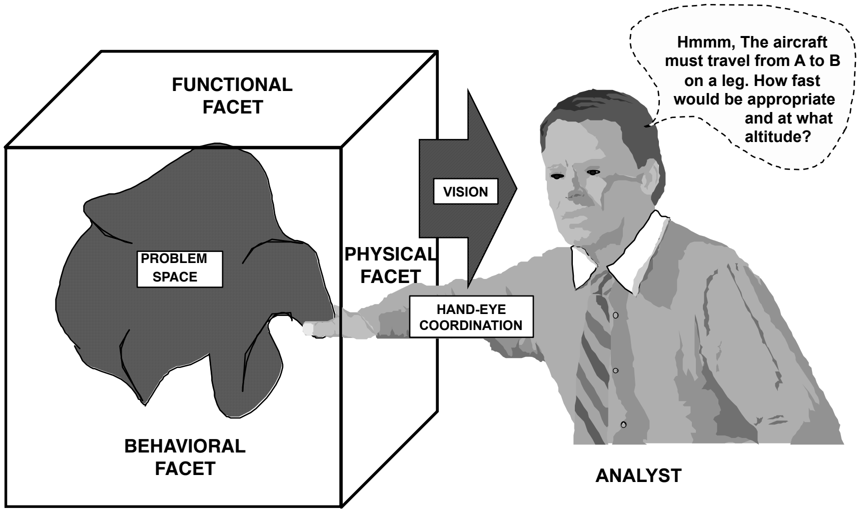
X: REFER TO PROGRAM SYSTEM DEFINITION DOCUMENT FOR EXPANSION

Major Problem on All Programs - Specification Content

- Each specification contains the essential characteristics its product or interface entity must possess in the form of requirements
- An enterprise should derive the content of all specifications on all programs using a single comprehensive universal architecture description framework (UADF) model
  - Functional
  - MSA-PSARE
  - UML-SysML
  - UPDM maybe
- Adopt the Model-RAS-Specification Sequence using your selected UADF and a template coordinated with it

Models Channel Requirements Into the Human Mind Through Vision –

A Picture is Worth 10^3 Words

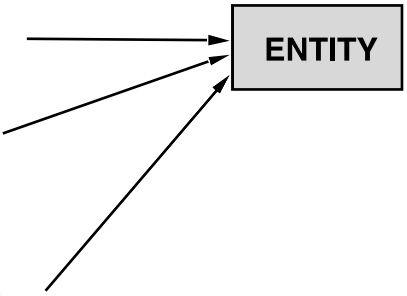

FUNCTIONAL FACET
PROBLEM SPACE
VISION
PHYSICAL FACET
HAND-EYE COORDINATION
BEHAVIORAL FACET
Hmmm, The aircraft must travel from A to B on a leg. How fast would be appropriate and at what altitude?
ANALYST

The First Objective of Modeling - Architecture

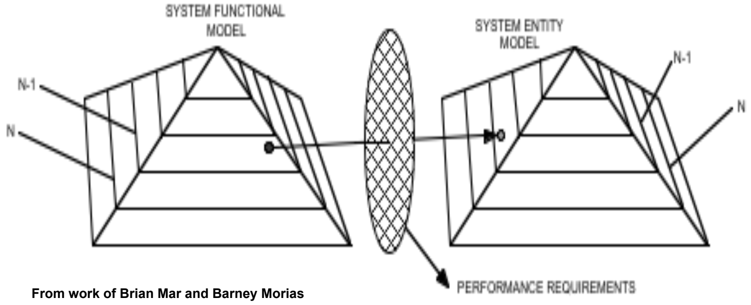
- What mission objective does the customer wish to achieve?
- What product entities shall the system consist of?
- How shall those product entities be inter-related through interfaces?
- What does the system environment consist of?
- How are the product entities related to the environment?
- What specialty engineering domains must be respected in the design?

The Second Objective of Modeling - Requirements

- ● Something wanted or necessary.
- ● Something essential to the existence or occurrence of an entity.
- ● A necessary character-istic or attribute of some thing (or entity).



Progressive Modeling

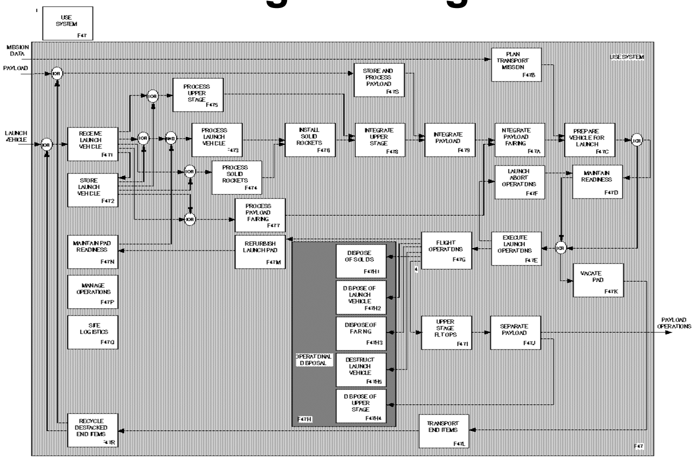


From work of Brian Mar and Barney Morias

Three UADF Are Available

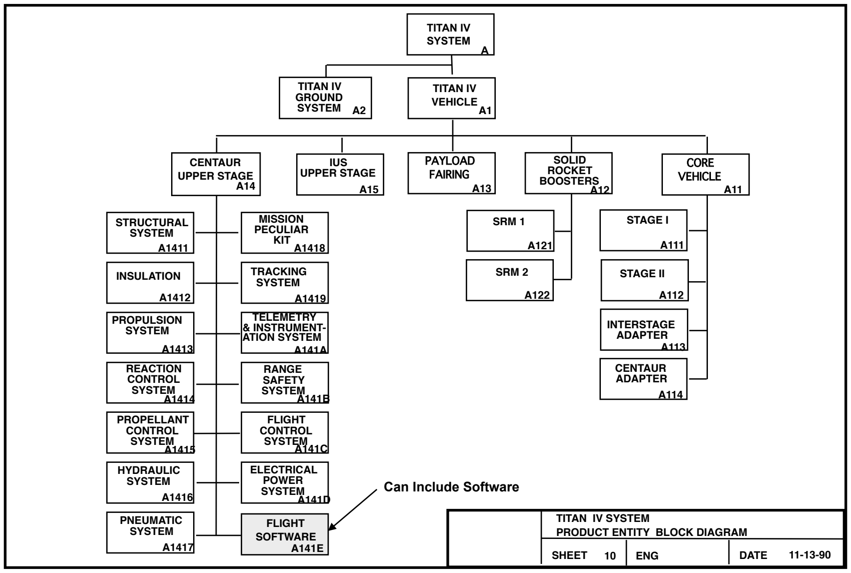
- A UADF is a comprehensive modeling approach in that it matters not how you will implement the solution in HW, SW, or people doing things
- One model is equally effective in HW and SW
- Pick one
  - Functional
  - MSA-PSARE
  - UML-SysML
  - UPDM maybe

Functional UADF Functional Flow Diagramming


USE SYSTEM
F47
MISSION DATA
PAYLOAD
LAUNCH VEHICLE
IOR
AND
USE SYSTEM
RECEIVE LAUNCH VEHICLE F471
STORE LAUNCH VEHICLE F472
PROCESS LAUNCH VEHICLE F473
PROCESS SOLID ROCKETS F474
PROCESS UPPER STAGE F475
INSTALL SOLID ROCKETS F476
PROCESS PAYLOAD FAIRING F477
INTEGRATE UPPER STAGE F478
INTEGRATE PAYLOAD F479
INTEGRATE PAYLOAD FAIRING F47A
PLAN TRANSPORT MISSION F47B
PREPARE VEHICLE FOR LAUNCH F47C
MAINTAIN READINESS F47D
EXECUTE LAUNCH OPERATIONS F47E
LAUNCH ABORT OPERATIONS F47F
FLIGHT OPERATIONS F47G
OPERATIONAL DISPOSAL
DISPOSE OF SOLIDS F47H1
DISPOSE OF LAUNCH VEHICLE F47H2
DISPOSE OF FAIRING F47H3
DISPOSE OF UPPER STAGE F47H4
DESTRUCT LAUNCH VEHICLE F47H5
F47H
UPPER STAGE FLT OPS F47I
SEPARATE PAYLOAD F47J
VACATE PAD F47K
TRANSPORT END ITEMS F47L
REFURBISH LAUNCH PAD F47M
MAINTAIN PAD READINESS F47N
MANAGE OPERATIONS F47P
SITE LOGISTICS F47Q
RECYCLE DESTACKED END ITEMS F47R
STORE AND PROCESS PAYLOAD F47S
PAYLOAD OPERATIONS
F47

But this technique will work with any UADF.

Functional UADF Product Entity Diagram


TITAN IV SYSTEM A

TITAN IV GROUND SYSTEM A2

TITAN IV VEHICLE A1

CENTAUR UPPER STAGE A14

IUS UPPER STAGE A15

PAYLOAD FAIRING A13

SOLID ROCKET BOOSTERS A12

CORE VEHICLE A11

STRUCTURAL SYSTEM A1411

INSULATION A1412

PROPULSION SYSTEM A1413

REACTION CONTROL SYSTEM A1414

PROPELLANT CONTROL SYSTEM A1415

HYDRAULIC SYSTEM A1416

PNEUMATIC SYSTEM A1417

MISSION PECULIAR KIT A1418

TRACKING SYSTEM A1419

TELEMETRY & INSTRUMENT-ATION SYSTEM A141A

RANGE SAFETY SYSTEM A141B

FLIGHT CONTROL SYSTEM A141C

ELECTRICAL POWER SYSTEM A141D

FLIGHT SOFTWARE A141E

SRM 1 A121

SRM 2 A122

STAGE I A111

STAGE II A112

INTERSTAGE ADAPTER A113

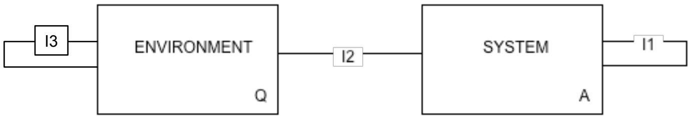
CENTAUR ADAPTER A114

Can Include Software

TITAN IV SYSTEM
PRODUCT ENTITY BLOCK DIAGRAM

SHEET 10 ENG DATE 11-13-90

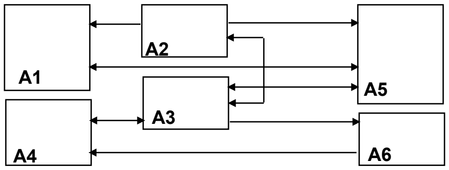
Functional UADF
Top-Level View of System Interface



Internal Interface
I1 Innerface
External Interface
I2 Crossface
I3 Outerface

Functional UADF
Two Interface Reporting Models

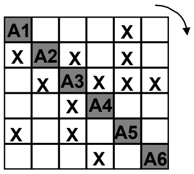
Schematic block diagramming



Lines define interfaces

Blocks are objects selected only from the product entity structure

N-square diagramming



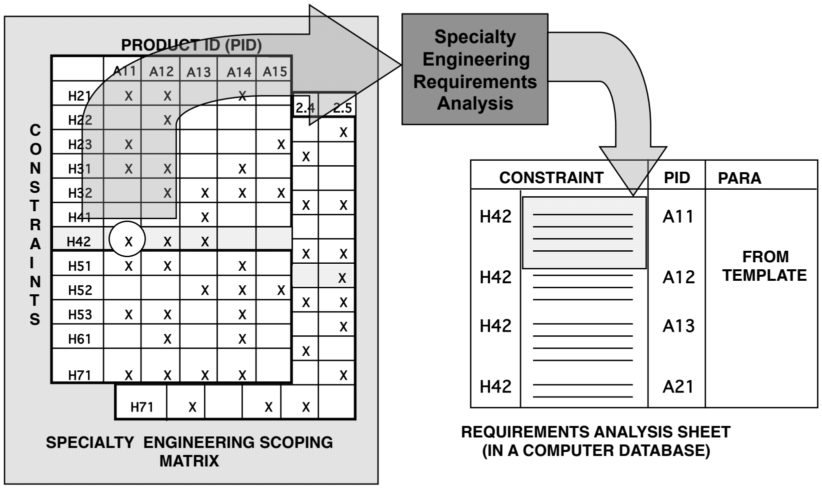
A1				X	
X	A2	X		X	
	X	A3	X	X	X
		X	A4		
X		X		A5	
			X		A6

Marked intersections define interfaces

Diagonal blocks are objects only from product entity structure

Apparent ambiguity reflects directionality

Functional UADF
Specialty Engineering Scoping Matrix

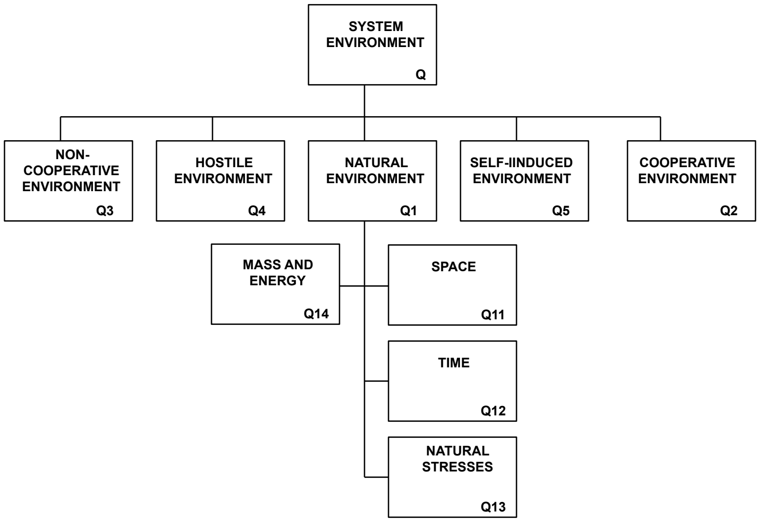

PRODUCT ID (PID)
CONSTRAINTS
2.4
2.5
Specialty Engineering Requirements Analysis
SPECIALTY ENGINEERING SCOPING MATRIX
REQUIREMENTS ANALYSIS SHEET (IN A COMPUTER DATABASE)

	A11	A12	A13	A14	A15
H21	X	X		X	
H22		X			
H23	X				X
H31	X	X		X	
H32		X	X	X	X
H41			X		
H42	X	X	X		
H51	X	X		X	
H52			X	X	X
H53	X	X		X	
H61		X		X	
H71	X	X	X	X	

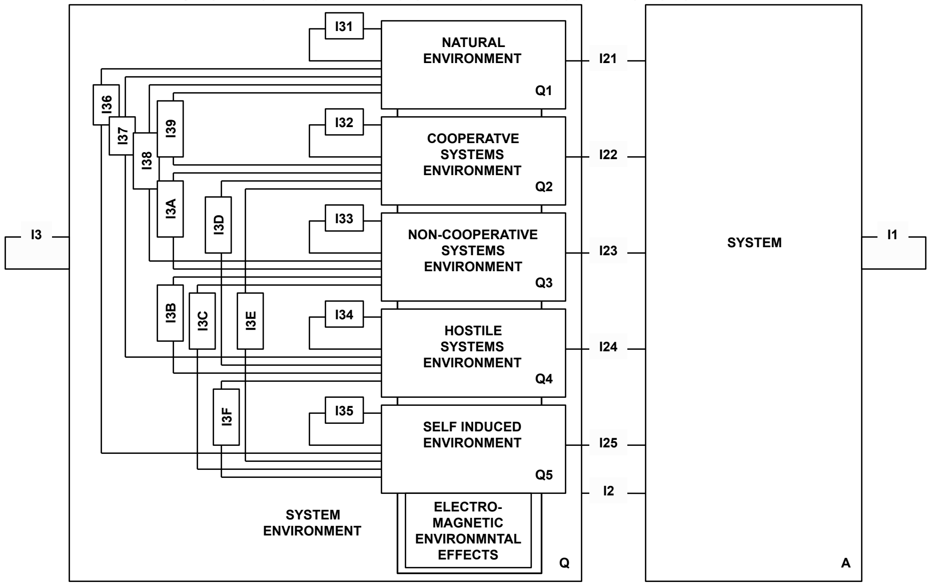
CONSTRAINT	PID	PARA
H42	A11	FROM TEMPLATE
H42	A12	
H42	A13	
H42	A21	

REQUIREMENTS ANALYSIS SHEET
(IN A COMPUTER DATABASE)

Functional UADF
Environmental Classes



Functional UADF
Generic External Interface MID

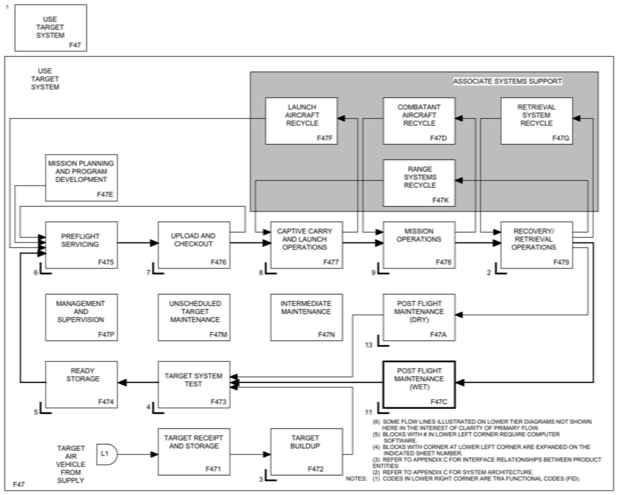


Functional UADF
Three Tier Environmental Modeling

- System level using integrated union of tailored standards
- End item level using three dimensional service use profile
  - Product entities
  - Environmental stresses
  - Process steps
- Component level using end item zoning and mapping components to zones
- Possible need for an environmental sub system

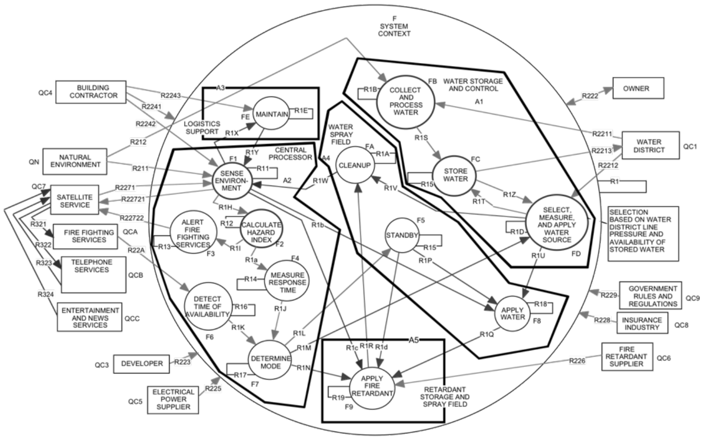
Functional UADF Process Flow Diagram

Needed as Part of the End Item Environmental Model


1
USE TARGET SYSTEM F47
USE TARGET SYSTEM
ASSOCIATE SYSTEMS SUPPORT
LAUNCH AIRCRAFT RECYCLE F47F
COMBATANT AIRCRAFT RECYCLE F47D
RETRIEVAL SYSTEM RECYCLE F47G
RANGE SYSTEMS RECYCLE F47K
MISSION PLANNING AND PROGRAM DEVELOPMENT F47E
PREFLIGHT SERVICING F475
6
UPLOAD AND CHECKOUT F476
7
CAPTIVE CARRY AND LAUNCH OPERATIONS F477
8
MISSION OPERATIONS F478
9
RECOVERY/ RETRIEVAL OPERATIONS F479
2
MANAGEMENT AND SUPERVISION F47P
UNSCHEDULED TARGET MAINTENANCE F47M
INTERMEDIATE MAINTENANCE F47N
POST FLIGHT MAINTENANCE (DRY) F47A
13
READY STORAGE F474
5
TARGET SYSTEM TEST F473
4
POST FLIGHT MAINTENANCE (WET) F47C
11
TARGET AIR VEHICLE FROM SUPPLY
L1
TARGET RECEIPT AND STORAGE F471
TARGET BUILDUP F472
3
F47

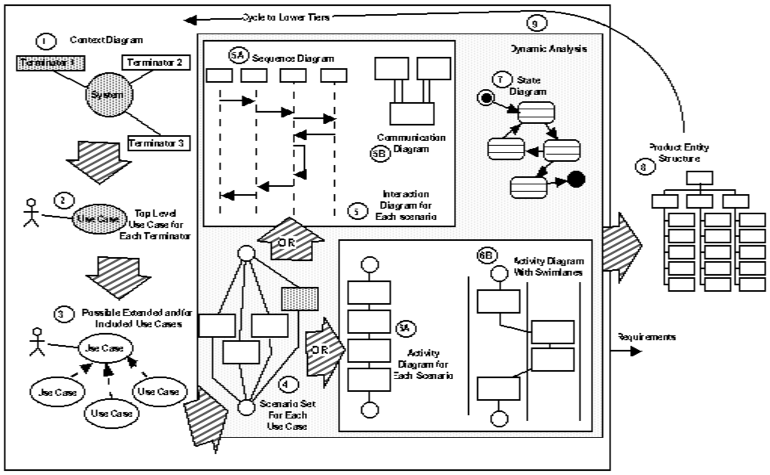
NOTES:
(6) SOME FLOW LINES ILLUSTRATED ON LOWER TIER DIAGRAMS NOT SHOWN HERE IN THE INTEREST OF CLARITY OF PRIMARY FLOW.
(5) BLOCKS WITH # IN LOWER LEFT CORNER REQUIRE COMPUTER SOFTWARE.
(4) BLOCKS WITH CORNER AT LOWER LEFT CORNER ARE EXPANDED ON THE INDICATED SHEET NUMBER.
(3) REFER TO APPENDIX C FOR INTERFACE RELATIONSHIPS BETWEEN PRODUCT ENTITIES
(2) REFER TO APPENDIX C FOR SYSTEM ARCHITECTURE.
(1) CODES IN LOWER RIGHT CORNER ARE TRA FUNCTIONAL CODES (FID).

Systems Development Using the MSA-PSARE UADF



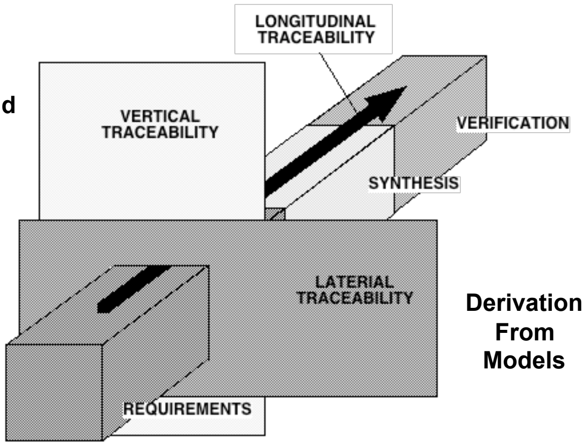
Assign Product Entity MID (A) to Super Bubbles
Assign Interface MID (I) to Functional Relations (R)

System Development Using the UML-SysML UADF

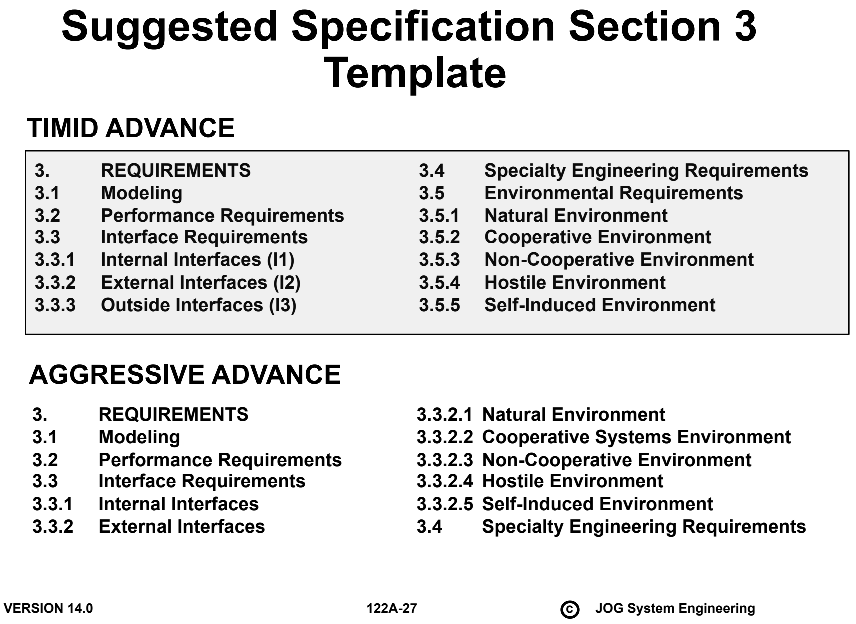

Cycle to Lower Tiers
9
1 Context Diagram
Terminator 1
Terminator 2
System
Terminator 3
2
Use Case
Top Level Use Case for Each Terminator
3 Possible Extended and/or Included Use Cases
Use Case
Use Case
Use Case
Use Case
5A Sequence Diagram
Communication Diagram 5B
Interaction Diagram for Each scenario 5
OR
OR
4 Scenario Set For Each Use Case
6A
Activity Diagram for Each Scenario
6B Activity Diagram With Swimlanes
Dynamic Analysis
7 State Diagram
Product Entity Structure 8
Requirements

No Matter the UADF Selected – Employ Three-Dimensional Requirements Traceability

Parent-Child
Source
Rationale



Derivation
From
Models



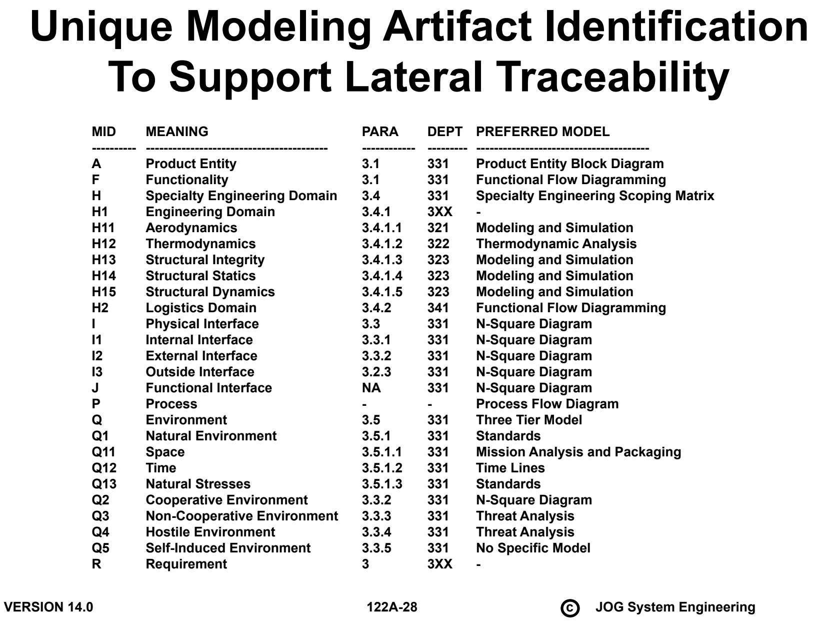
Suggested Specification Section 3 Template

TIMID ADVANCE

3. REQUIREMENTS
3.1 Modeling
3.2 Performance Requirements
3.3 Interface Requirements
3.3.1 Internal Interfaces (I1)
3.3.2 External Interfaces (I2)
3.3.3 Outside Interfaces (I3)
3.4 Specialty Engineering Requirements
3.5 Environmental Requirements
3.5.1 Natural Environment
3.5.2 Cooperative Environment
3.5.3 Non-Cooperative Environment
3.5.4 Hostile Environment
3.5.5 Self-Induced Environment

AGGRESSIVE ADVANCE

3. REQUIREMENTS
3.1 Modeling
3.2 Performance Requirements
3.3 Interface Requirements
3.3.1 Internal Interfaces
3.3.2 External Interfaces
3.3.2.1 Natural Environment
3.3.2.2 Cooperative Systems Environment
3.3.2.3 Non-Cooperative Environment
3.3.2.4 Hostile Environment
3.3.2.5 Self-Induced Environment
3.4 Specialty Engineering Requirements



Unique Modeling Artifact Identification To Support Lateral Traceability

MID	MEANING	PARA	DEPT	PREFERRED MODEL
A	Product Entity	3.1	331	Product Entity Block Diagram
F	Functionality	3.1	331	Functional Flow Diagramming
H	Specialty Engineering Domain	3.4	331	Specialty Engineering Scoping Matrix
H1	Engineering Domain	3.4.1	3XX	-
H11	Aerodynamics	3.4.1.1	321	Modeling and Simulation
H12	Thermodynamics	3.4.1.2	322	Thermodynamic Analysis
H13	Structural Integrity	3.4.1.3	323	Modeling and Simulation
H14	Structural Statics	3.4.1.4	323	Modeling and Simulation
H15	Structural Dynamics	3.4.1.5	323	Modeling and Simulation
H2	Logistics Domain	3.4.2	341	Functional Flow Diagramming
I	Physical Interface	3.3	331	N-Square Diagram
I1	Internal Interface	3.3.1	331	N-Square Diagram
I2	External Interface	3.3.2	331	N-Square Diagram
I3	Outside Interface	3.2.3	331	N-Square Diagram
J	Functional Interface	NA	331	N-Square Diagram
P	Process	-	-	Process Flow Diagram
Q	Environment	3.5	331	Three Tier Model
Q1	Natural Environment	3.5.1	331	Standards
Q11	Space	3.5.1.1	331	Mission Analysis and Packaging
Q12	Time	3.5.1.2	331	Time Lines
Q13	Natural Stresses	3.5.1.3	331	Standards
Q2	Cooperative Environment	3.3.2	331	N-Square Diagram
Q3	Non-Cooperative Environment	3.3.3	331	Threat Analysis
Q4	Hostile Environment	3.3.4	331	Threat Analysis
Q5	Self-Induced Environment	3.3.5	331	No Specific Model
R	Requirement	3	3XX	-

RAS-Complete In Table Form

MODEL ENTITY		REQUIREMENT ENTITY		PRODUCT ENTITY		DOCUMENT ENTITY	
MID	MODEL ENTITY NAME	RID	REQUIREMENT	PID	ITEM NAME	PARA	TITLE
F47	Use System			A	Product System		
F471	Deployment Ship Operations			A	Product System		
F4711	Store Array Operationally	RXR67	Storage Volume < 10 ISO Vans	A1	Sensor Subsystem		
H	Specialty Engineering Disciplines			A	Product System		
H11	Reliability	REW34	Failure Rate < 10 x 10-6	A1	Sensor Subsystem	3.1.5	Reliability
H11	Reliability	RG31R	Failure Rate < 3 x 10-6	A11	Cable	3.1.5	Reliability
H11	Reliability	RFYH4	Failure Rate < 5 x 10-6	A12	Sensor Element	3.1.5	Reliability
H11	Reliability	RG8R4	Failure Rate < 2 x 10-6	A13	Pressure Vessel	3.1.5	Reliability
H12	Maintainability	R6GHU	Mean Time to Repair < 0.2 Hours	A1	Sensor Subsystem	3.1.6	Maintainability
H12	Maintainability	RU9R4	Mean Time to Repair < 0.4 Hours	A11	Cable	3.1.6	Maintainability
H12	Maintainability	RJ897	Mean Time to Repair < 0.2 Hours	A12	Sensor Element	3.1.6	Maintainability
H12	Maintainability	R9D7H	Mean Time to Repair < 0.1 Hours	A13	Pressure Vessel	3.1.6	Maintainability
I	System Interface			A	Product System		
I1	Internal Interface			A	Product System		
I11	Sensor Subsystem Innerface			A1			
I181	Aggregate Signal Feed Source Impedance	RE37H	Aggregate Signal Feed Source Impedance= 52 ohms ± 2 ohms	A1	Sensor Subsystem		
I181	Aggregate Signal Feed Load Impedance	RE37I	Aggregate Signal Feed Load Impedance= 52 ohms ± 2 ohms	A4	Analysis and Reporting Subsystem		
I2	System External Interface			A	Product System		
Q	System Environment			A	Product System		
QH	Hostile Environment			A	Product System		
QI	Self-Induced Environmental Stresses			A	Product System		
QN	Natural Environment			A	Product System		
QN1	Temperature	R6D74	-40 degrees F< Temperature < +140 degrees F	A	Product System		
QX	Non-Cooperative Environmental Stresses			A	Product System		

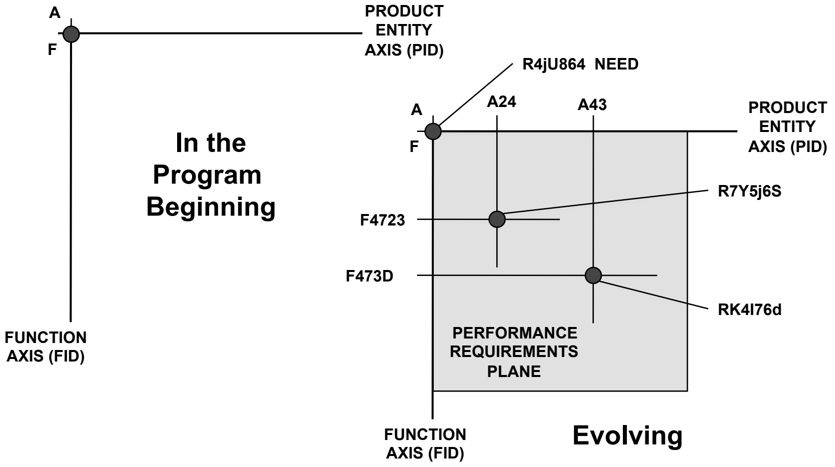
The Requirements Analysis Sheet (RAS)

- **Tabular RAS in a computer database from which specifications may be printed is needed on every program**
- **Graphical RAS will be used in this presentation to explain the content and loading the tabular RAS from models**
- **In this presentation the functional UADF modeling artifacts are used in building the graphical RAS but the idea is compatible with the other two UADF as well**

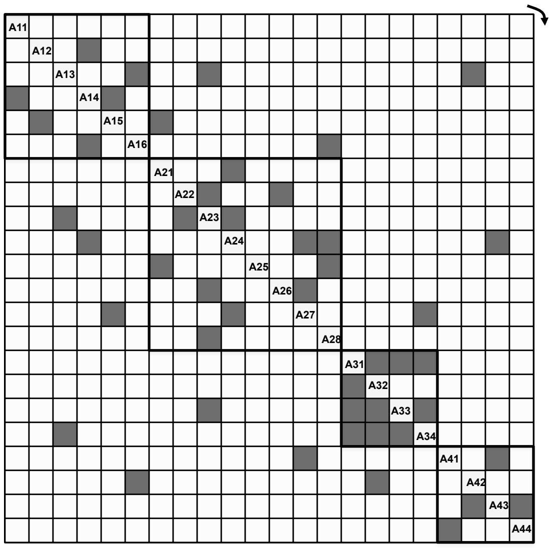
Capture the Model and Configuration Manage It

- **Systems Architecture Report (SAR) Recommended**
- **For the Functional UADF the following appendices are suggested**
  - **A Functional Flow Diagram**
  - **B Environment (Natural, Cooperative, Non-cooperative, Hostile, Self-Induced)**
  - **C Product Entity Block Diagram**
  - **D Interface Diagram (Schematic Block or N-Square Diagram)**
  - **E Specialty Engineering Scoping Matrix**
  - **F Process Diagram**
  - **G RAS or reference to its location**

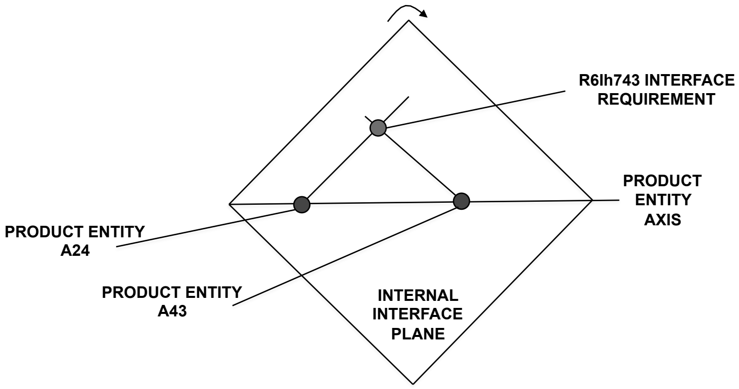
Graphical RAS – Performance Requirements Plane



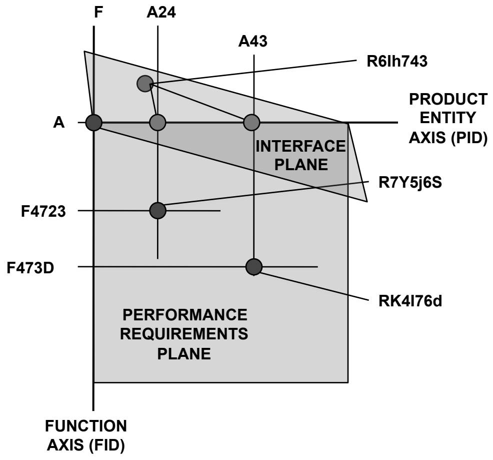
Graphical RAS – Internal Interface Plane



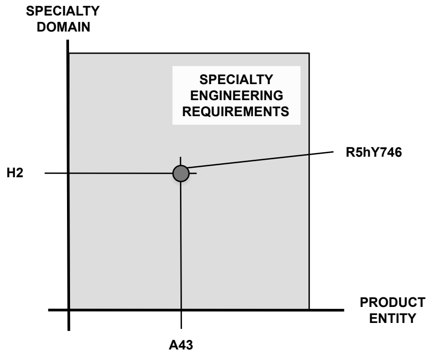
Graphical RAS – Rotate Internal Interface Plane



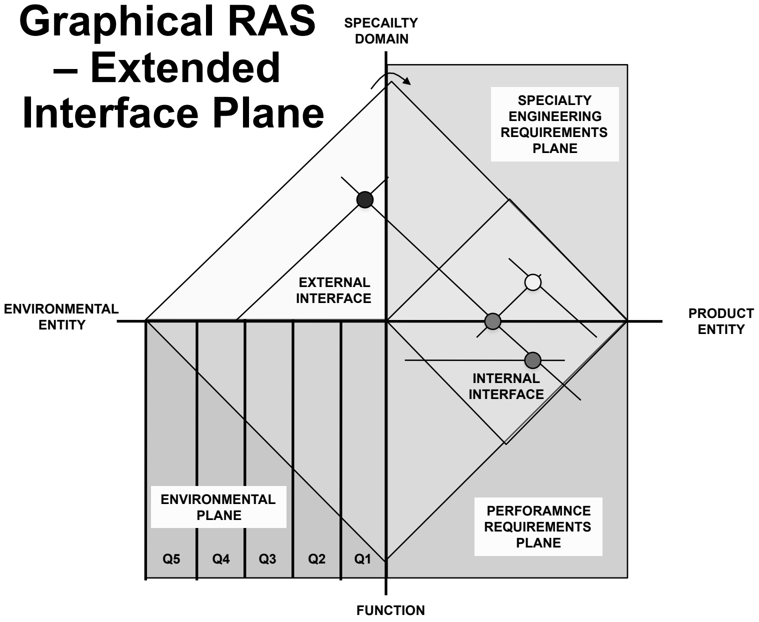
Graphical RAS – Functional Plane Coordinated With Interface Plane



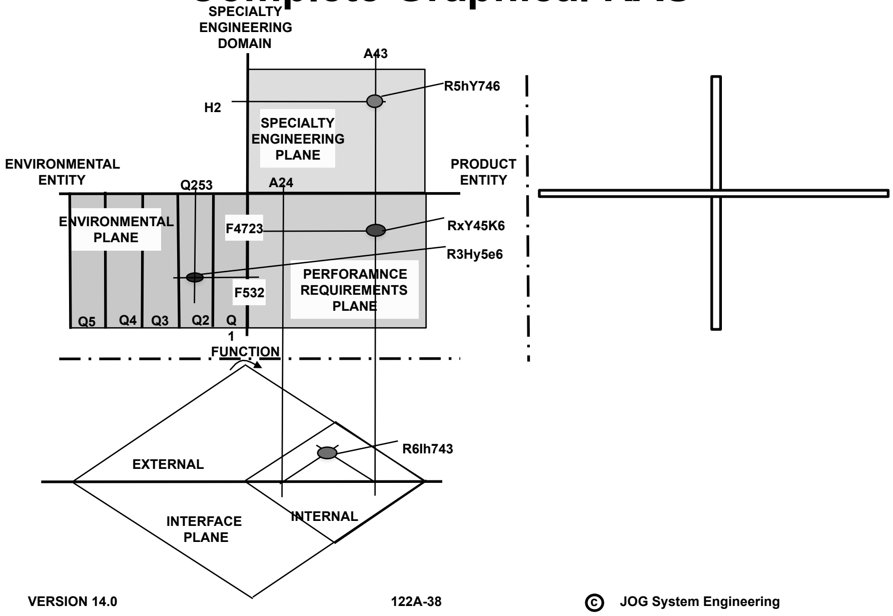
Graphical RAS – Specialty Engineering Plane



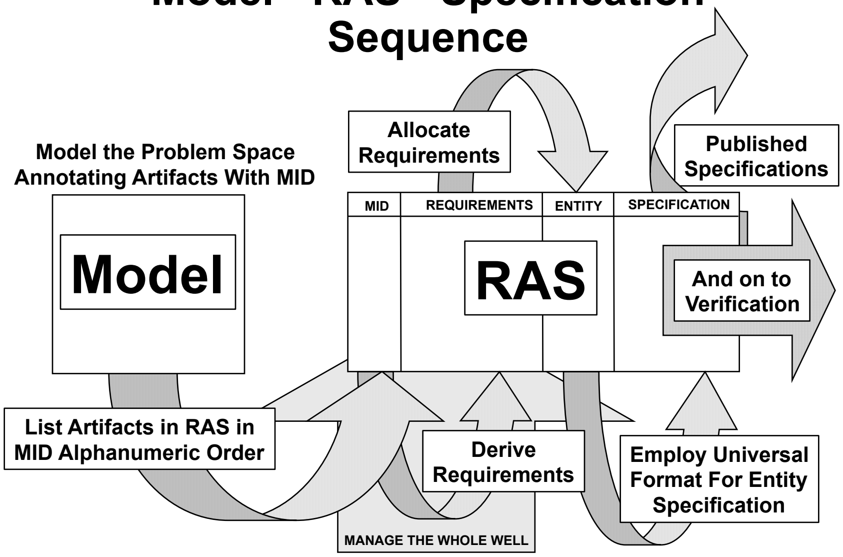
Graphical RAS – Extended Interface Plane



Complete Graphical RAS



Model - RAS - Specification Sequence



Prescription For the Enterprise That Has Not Yet Reached Perfection

1. Adopt a UADF and insist that all persons doing architecture development and requirements analysis work use it.
2. Adopt a way of uniquely identifying all modeling artifacts from which requirements may be derived.
3. Adopt a means by which personnel may capture modeling and specification content such that they may be configuration managed. There are not any computer tools known to the author that could capture all of the modeling and documentation features covered in the paper but one could build a simple text-oriented database linked to hand drawn or computer application graphics modeling artifacts.
4. Adopt a means for personnel to accomplish modeling work and retention of masters in the formal system baseline documentation.
5. Adopt a set of specification templates coordinated with modeling.
6. Establish a policy such as Table 1 of the supporting text suggests that clearly assigns responsibility for all specification content to personnel from specific functional departments on all programs.
7. Prepare a written document telling how this work is to be done on programs.
8. Train all personnel who have a role in this work in the appropriate parts of it assigned to their functional department.
9. Establish a quality assurance means that will assure that the work is accomplished in accordance with the prepared instructions and contractual requirements on programs.