JOG SYSTEM ENGINEERING GRAND SYSTEMS DEVELOPMENT TRAINING PROGRAM PRESENTATION

VERSION 13.0

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

FROM WHERE WE ARE TO WHERE WE ARE GOING IN SYSTEMS ENGINEERING VOLUME 322-12B

We have a past and despite what some people say the profession has a bright future

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

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

From Where We Are to Where We Are Going in Systems Engineering

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

1. Overview

Many have concluded that the profession of systems engineering is dull, overly focused on order and discipline, and devoid of enjoyable work. Some or all of these characteristics may have been valid criticisms in our past but those who practice this art are on the verge of a tremendous leap forward that guarantees a larger population of system engineers enjoying increasingly fascinating work. Before offering the story of the future let us chart where we are now after many starts and stops to improve the process of systems development.

2. Where We Are

2.1 A Long Time Coming

Over time on planet Earth mankind has amassed more knowledge than can be contained in a single human mind, as suggested in Figure 1, by individual human beings specializing in some subset of knowledge in combination with evolving effective ways for individuals to cooperate and communicate about a shared problem. The result has been the evolution of specialists who are deeply but very narrowly focused in some single knowledge domain. The profession of system engineering evolved as a means for these narrow specialists to effectively communicate across cross-product and cross-domain boundaries and act collectively as the equivalent on one great mind. An ever-increasing knowledge expanse available to mankind continues to open new vistas for the recognition of new and more difficult problems needing solution and insights into their solution. There is, therefore, a natural bias for an increasing population of system engineers in well-managed enterprises involved in the pursuit of solutions for very difficult problems.

Older domain engineers can recall a time when they worked out solutions to very difficult problems through careful thought and hand drawn sketches aided by a slide rule The system development work accomplished by engineers on programs has for several years, since the invention of the digital computer, been undergoing a significant change that is now approaching a dramatic revolution referred to as model-based system engineering. In this new world the specialized engineers will continue interacting with domain-specific models on engineering work stations but behind the screen they are looking at there will be a much more intense machine action taking place than at present. As a result, the system engineer will have to continue to deal with the cross-product and cross-domain communication problems but add the cross-model problem as well. Increasing complexity is the playground of the system engineer so the future promises to be an enjoyable one for those adjusted to the changes and prepared to participate.

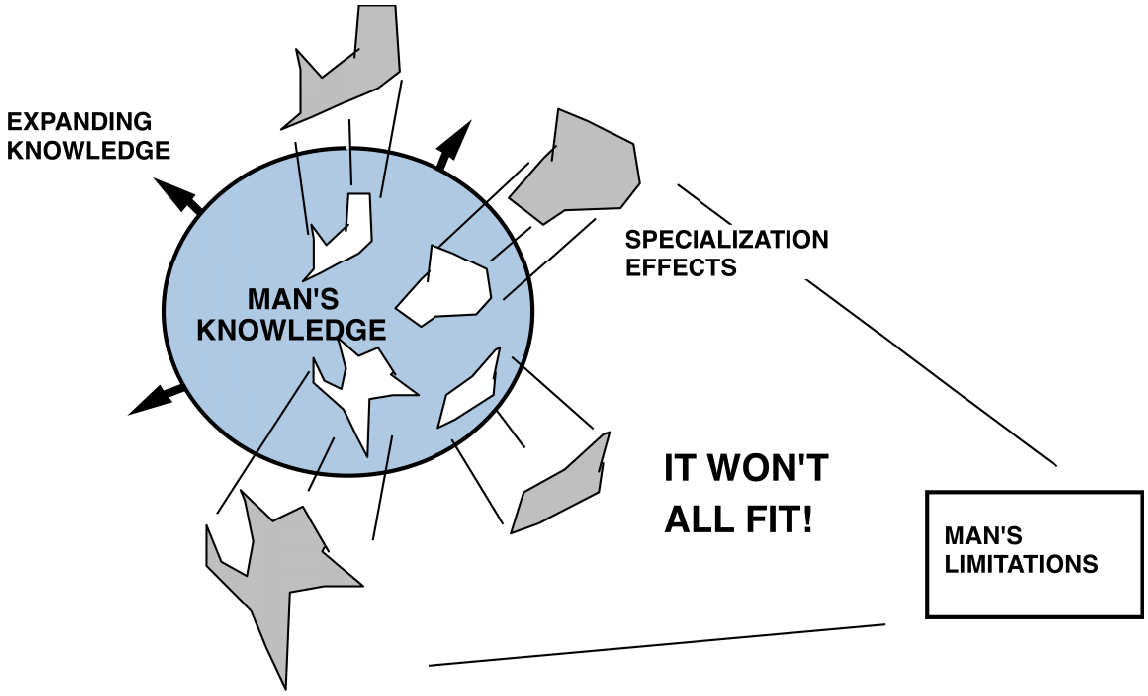


Figure 1 Mankind's Knowledge Base Far Exceeds Individual Man's Capacity

As Robert E. Machol said in the opening chapter of "System Engineering Handbook", published by McGraw-Hill in 1965 and for which he was the editor, that the mind of a system engineer was "T-shaped" as shown in Figure 2.

Actually, all of us humans, are built this way. Thousands of years ago mankind passed through a milestone when the aggregate of knowledge available to mankind became greater then the individual human mental capacity for knowledge. At that time the profession of system engineering was pre-determined. Knowledge has value in solving problems so it was natural that mankind would follow the path that included individual specialization demanding more effective human communication to bridge the gaps between the specialists. The specialists will ever have some trouble reaching agreement among themselves and will often need the help of people who are interested in the breadth of knowledge axis on Figure 2. These people tend to collect in the profession of system engineering and it is the responsibility of system engineers to work toward causing the population of a team or a program to act as a single person with one great mind. Knowledge will continue to expand causing narrower and deeper specialists and therefore more cross-domain boundaries in the development of future systems.

The systems that are developed today are quite complex involving many different technological domains such as structures, electronics, fluids, computer software, aerodynamic flight shapes, and satellite communications. One could reasonably conclude that since we humans have solved all of the simple problems on Earth and that we now stand on a platform of solutions that permits us to see further into problems never before imagined, that the problems we undertake in the future will be more complex than those in the past and that these problems will involve more cross product boundary conditions than in the past.

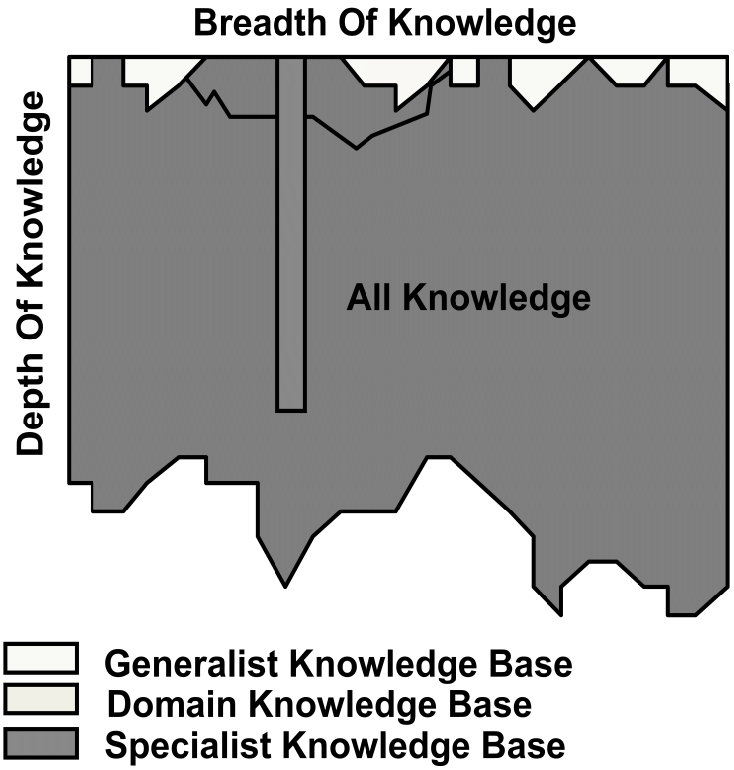


Figure 2 The Generalist Versus the Specialist

2.2 Enterprise Fundamentals

A given system development enterprise will implement and pursue many development programs over time and in some of these enterprises multiple programs will be in progress simultaneously as suggested in Figure 3. A well-managed enterprise exists within an environment of continual expansion of knowledge mastered by employees permitting the enterprise to pursue increasingly complex programs. This points up the value of knowledge and it is in fact the most valuable commodity the enterprise possesses. Therefore, the enterprise must organize itself so as to ensure that this knowledge can be brought to bear on program problems most effectively.

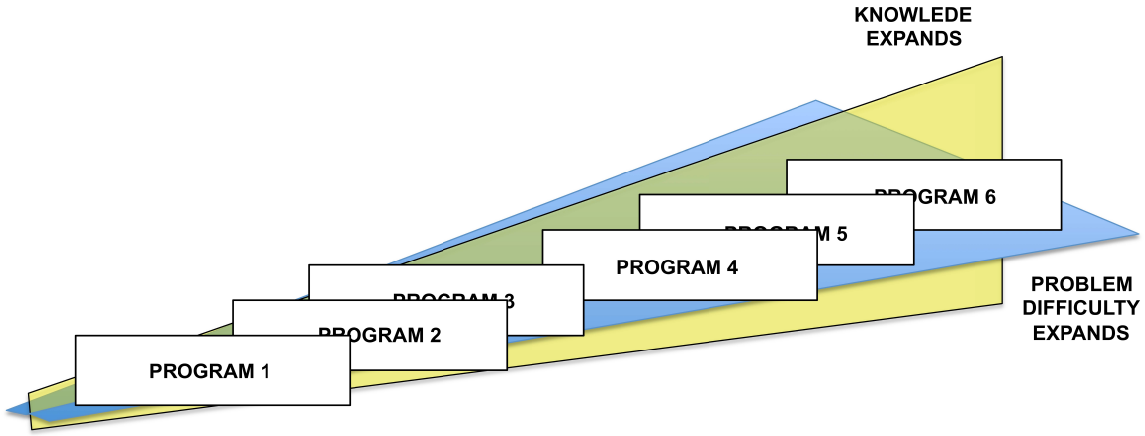


Figure 3 Expanding Capability Over Time

Each of the programs operated by an enterprise would, ideally, follow essentially the same common process leading to the employees benefitting over the long run from the training benefits of repetition. Figure 4 illustrates a common process for such an enterprise focused on recognizing that programs have to accomplish three fundamental development tasks in essentially the same order: (1) define the problem to be solved in a collection of specifications the content of which, ideally, was derived from models, (2) solve the problem defined in the specifications through design, procurement, and manufacture, and (3) verify that what is produced satisfies the requirements that drove the synthesis work. Further, each program must accomplish these three tasks within a sound management infrastructure applied across the whole program.

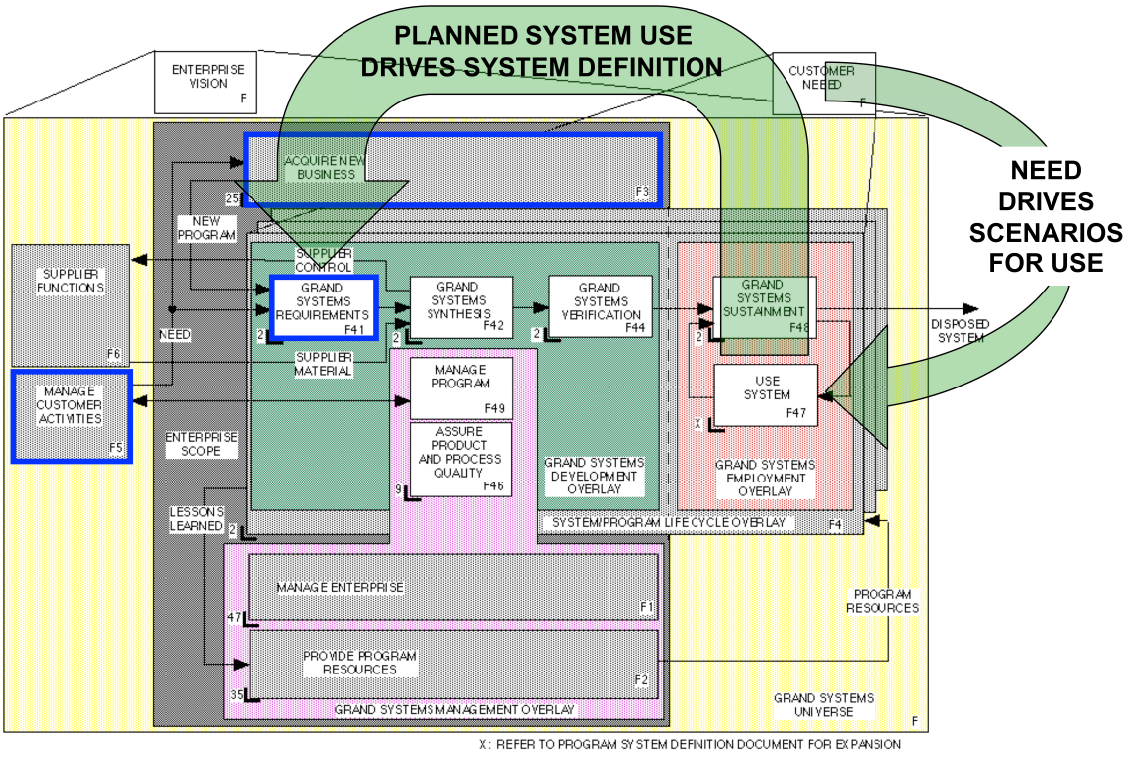


Figure 4 Program Common Process

The work of programs is accomplished by human beings in accordance with a plan and schedule managed within cost and schedule limits for customers often in accordance with a contract. Enterprises that wish to continue doing this work will work toward minimizing cost and execution time by bringing to bear the knowledge they have access to from their employees. This is most effectively accomplished if the available knowledge is organized in some understandable fashion.

Domain knowledge is best organized in an enterprise by identifying functional departments composed of personnel with domain knowledge. Each of these departments is led by a manager who understands the work of the department, how it is accomplished, and the relative skills of the individual members of the department in doing that work. Figure 5 illustrates the organizational structure of an enterprise that is managed through a matrix structure showing the functional departments in a kind of inverted "U". Each of these functional departments is a home

room for its members but should not be the structure within which these members work on a program. Each program should be organized into product-oriented cross-functional teams staffed from the functional departments as a function of available program budget and schedule and its need for particular domain knowledge and skills.

As suggested in Figure 5, the enterprise is responsible for providing programs with the resources they need to be successful. This includes people who know what they are doing, good practices ideally in the form of a common process implemented on every program with employees constantly improving performance through repetition, a work space, and good networks of tools continuing to improve over time. These efforts on the part of the enterprise can be effective if the enterprise is well managed by people who carry on a long term plan for the betterment of the enterprise and its programs rather than an opportunity to temporarily express their ego for self-aggrandizement and continued motion up the ladder. A positive slope growth path is better every day than a saw tooth full of starts and restarts.

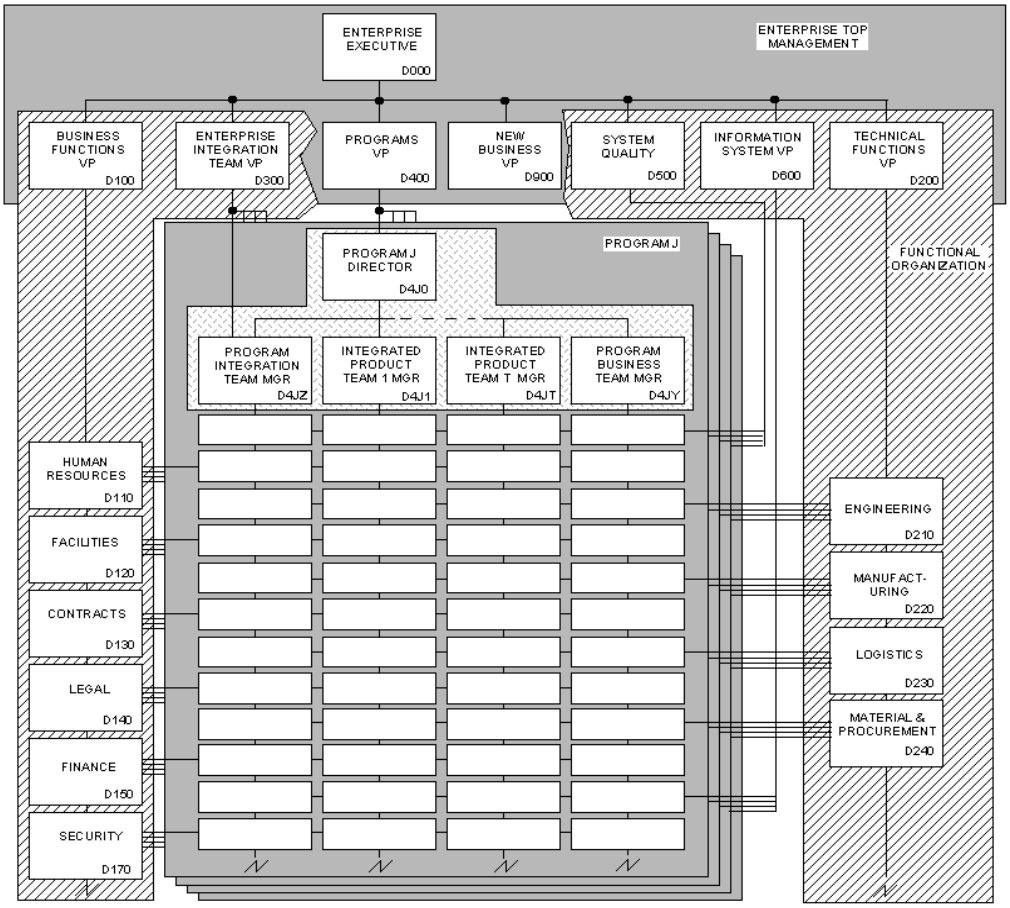


Figure 5 An Effective Enterprise Structure

The enterprise is best managed through a matrix where the enterprise collects employees into domain-oriented functional departments the managers of which focus on improving the performance of their specialty on programs as members of cross-functional teams oriented toward the product entity structure determined appropriate in early program work. The knock against matrix management in many enterprises is the difficulty in causing one or more managers

to function cooperatively because of the distance to a common superior. The reader will note in the author's view there should be a programs leader (VP in the diagram) as well as a functions leader. Figure 5 actually shows two functional VPs probably because the author never liked the business people and the pair makes the diagram look more balanced. However the functions are organized, they pour resources into teams as a function of available program budgets and schedules determined by the programs and those programs manage their teams toward program goals. The reader will also note on Figure 5 the presence of integration teams in both the enterprise and the programs. These are the home of system engineers mainly who apply their skills to perfecting the enterprise over time and the product systems each program is developing.

2.3 Problem Space Modeling

We entered the decade of the 1950s with a single modeling method wrapped around functional analysis employing flow diagrams composed of function blocks from which performance requirements were derived and allocated to product entities. Additional models were employed by some to complete the design constraints story dealing with interface, specialty engineering, and environmental requirements. These requirements might be entered in a requirements analysis sheet but then would have to be typed again into paper documents called specifications. While good efforts were made to configuration control the content of the specifications little effort was made to establish formal traceability between the modeling artifacts and the requirements derived from them or to capture or configuration control the modeling artifacts.

While analog computers were used throughout WWII operating as mechanical, electrical, and electromechanical machines to solve mathematical and logical problems to produce firing orders, a new breed of machines were coming into being called digital computers that required not only electronic machinery to function but a new product type called computer software that directed the operation of these machines. Those responsible for software development discovered they could apply functional flow diagrams to analyze the functional needs of this new product and follow essentially the same development pattern as found effective on hardware.

Over time some very bright people discovered alternative software modeling methods while the system and hardware engineers remained with functional flow diagrams to the extent that they used any formal method dealing with the system problem space. As suggested in Figure 6 we have now passed through several decades of modeling excess starting with a single comprehensive function-oriented model, often violated or ignored in favor of an ad hoc approach, expanding into a plethora of models, some effective and some not so. With the exception of PSARE none of these models are comprehensive meaning that an enterprise would have to apply two different modeling approaches, one for hardware and systems and one for software fracturing the development process in a pretty fundamental way.

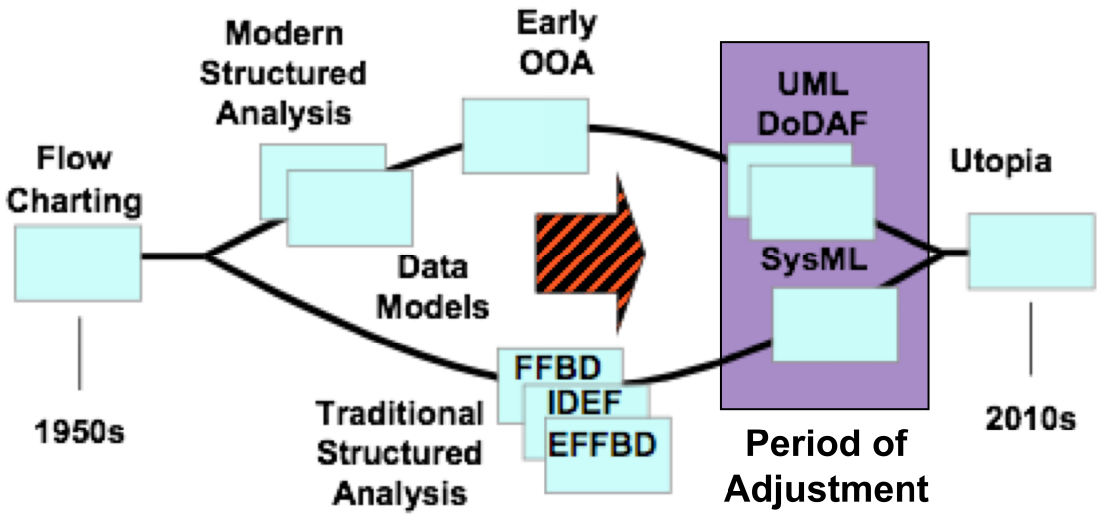


Figure 6 The History of Problem Space Modeling

Figure 6 suggests that we may be heading toward a return to a common comprehensive model for dealing with problem space and the author believes that is happening now. While there have been many models developed and we have been far less effective integrating and optimizing across this advancing front than we have been in creating the models, it is possible to collect them all into three sets of models referred to by the author as universal architecture description frameworks (UADF). Those three models are: (1) traditional structured analysis (TSA) composed of flow charting, product entity and interface block diagramming, specialty engineering scoping matrix and domain models, and a trio of environmental models; (2) modern structured analysis (MSA) combined with process for system architecture and requirements engineering (PSARE), the latter formally referred to as Hatley-Pirbhai or simply HP; and (3) unified modeling (UML) combined with system modeling language (SysML). An enterprise could today pick any of these model sets and complete a system problem space analysis that resulting in the capture of all requirements appearing in specifications having been derived from models, the models captured and configuration controlled, and the requirements propagated either in paper specifications or as the content of models contained within computer applications.

In that system engineers should be working toward improving communications across domain and product boundaries, this condition has led to serious problems for our profession. One solution to this problem is to link together a set of models that collectively is comprehensive. Functional analysis is one such model but it is unlikely that software people would support what they would see as retrogression to a past properly left behind. Two other comprehensive models are available, however, that have stronger support from the software community.

The combination of MSA and PSARE can be applied comprehensively if PSARE rules of permitting the bubbles of data flow diagrams to represent functionality accomplished by hardware or software and the directed line segments joining them may represent any kind of interface relationship rather than just data flow. Figure 7 illustrates the flow of information from the data flow diagram of MSA-PSARE into a list of modeling ID (MID) in a requirements analysis sheet captured in a computer database from which requirements are derived and allocated to product entities identified by overlaying super bubbles onto the DFD representing

product entities that will be responsible for accomplishing the functionality defined by the DFD bubbles enclosed. The requirements fall into the specifications for the entities ideally in a paragraph numbering order recognizing four types of requirements: (1) performance, (2) interface, (3) specialty engineering, and (4) environmental. Combine the constraints analysis of traditional structured analysis with MSA-PSARE or the architecture model of PSARE and the comprehensive model is complete.

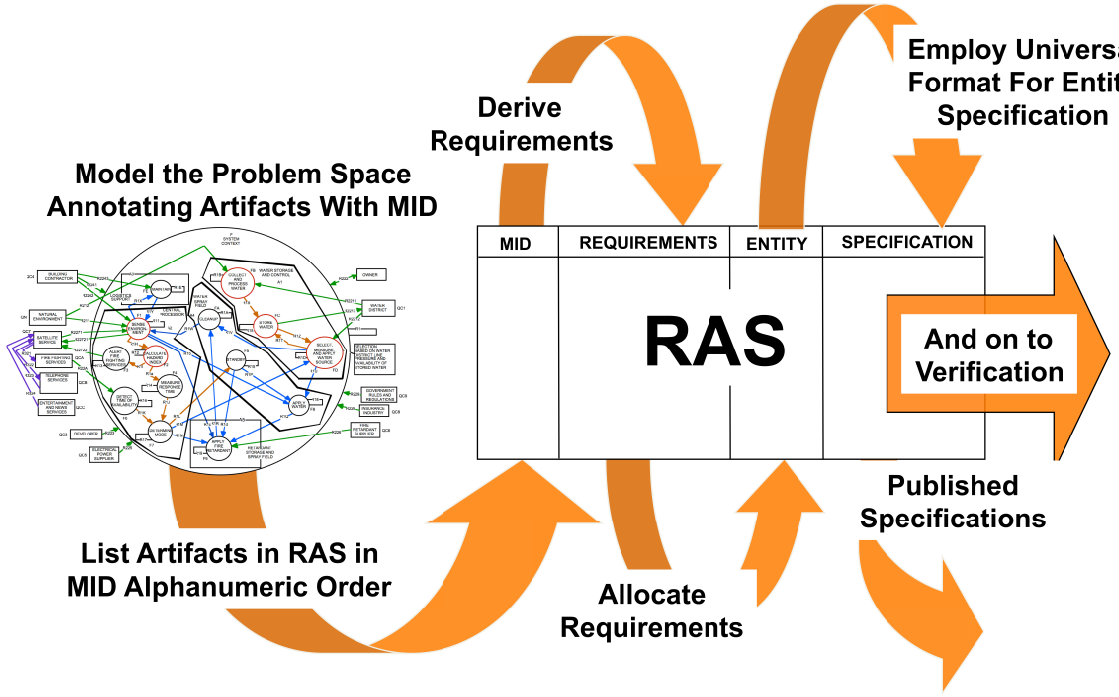


Figure 7 MSA-PSARE Modeling Flow

A third comprehensive model is realized from a combination of UML and system modeling language (SysML), selection of one of the two methods for product entity and interface identification, and extension to include specialty engineering and environmental modeling. Thus, we can today identify one comprehensive problem space model around which one could build effective computer applications to define problem space. In this model one identifies use cases from which scenarios evolve and are graphically examined through sequence and activity diagrams upon which exposed functionality and behavior are associated with product entities and interfaces identified.

Figure 8 illustrates an important characteristic of the three possible comprehensive models, referred to by the author as universal architecture description frameworks (UADF). They should support lateral traceability between the requirements and the models from which the requirements are derived and the such that there is a clear linkage throughout the problem space modeling that implementing computer applications can relate to. The use of models during problem space work is important because there is no existing reality so the engineers must probe the problem space using their imagination learning as they go about the solution to the problem. Figure 6 shows this linkage in terms of the use of the most powerful channel to pass information from outside the human mind into it, vision. The human apples hand-eye coordination to simple graphics to reflect thoughts about the problem space that in the process tends to dissipate

precipitating on the functional physical, and behavioral facets of the problem space model as simple sketches forming a model of the problem space.

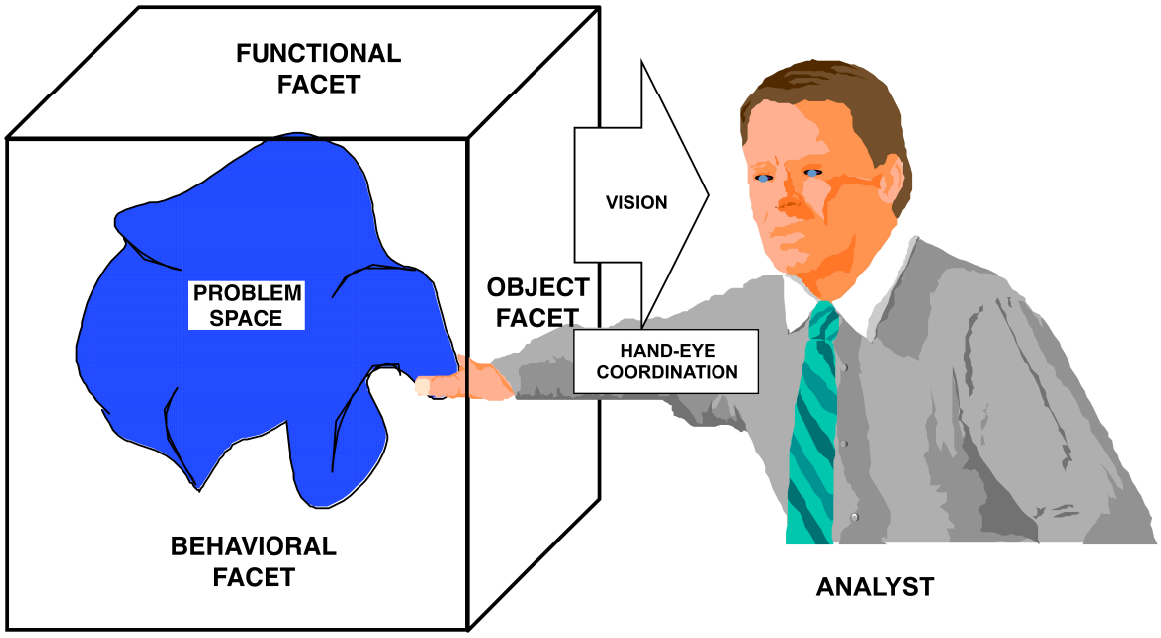


Figure 8 The Relationships Between Models and the Mind

Whatever problem space models are applied to day the results is captured in computer applications and today specifications are published from those applications.

2.4 Solution Space Modeling

As our knowledge about the problem space matures from models into the content of the specifications, the engineering specialists responsible for solving the problem apply their solution space models to determine appropriate and compliant features of the design. Mass properties engineers create mass models, reliability engineers create failure rate models, thermodynamics specialists, aerodynamicists, and structural dynamicists among others apply their models to craft a solution expressed in domain computer applications and engage in conversation while their conclusions derived from these modeling artifacts are questioned by their fellow specialists, system engineers, and managers.

There is a very intense conversation going on between those engineers accomplishing the problem space modeling and those doing the solution space modeling working from the product entity structure top downward to determine appropriate values of requirements included in the problem space models. Mass properties and reliability modeling offers a very simple example of this interaction but many more complex interactions are occurring as well.

This paper no doubt has so far fed the illusion that the synthesis process responding to clear knowledge of the problem will inevitably proceed in good order with discipline that would seem to be the goal of the system engineer but that is not intended. The design process is largely a creative activity that can involve considerable chaos and even an expression of passion on the part of those responsible for the results of their work when it is criticized.

It is during this very complicated "conversation" employing the worst interface on planet Earth, human communication, that many system development difficulties occur. The question is, "Can this conversation be automated to some degree and be improved in the process?" Will the design process become a simple expression of logical statements of the "If A then B" form where the human mind is not involved that can be completely accomplished by machines. What will be the right mix of dependence on human creativity and machine efficiency?

3. Where We Are Going

The author believes that Figure 9 represents how we will evolve relative to the use of organized problem space modeling beginning with document-driven development methods commonly implemented using typewriters. We are now emerging from a database-driven development approach where everyone has their own database and related computer applications acting as islands in the storm. Many people are experimenting with model-driven development that will employ a collection of interconnected models that can exchange information with less intense human attendance. All of this evolution has been happening against a backdrop of increasing use of a set of comprehensive problem space models closing on what the author refers to as the UML-SysML UADF.

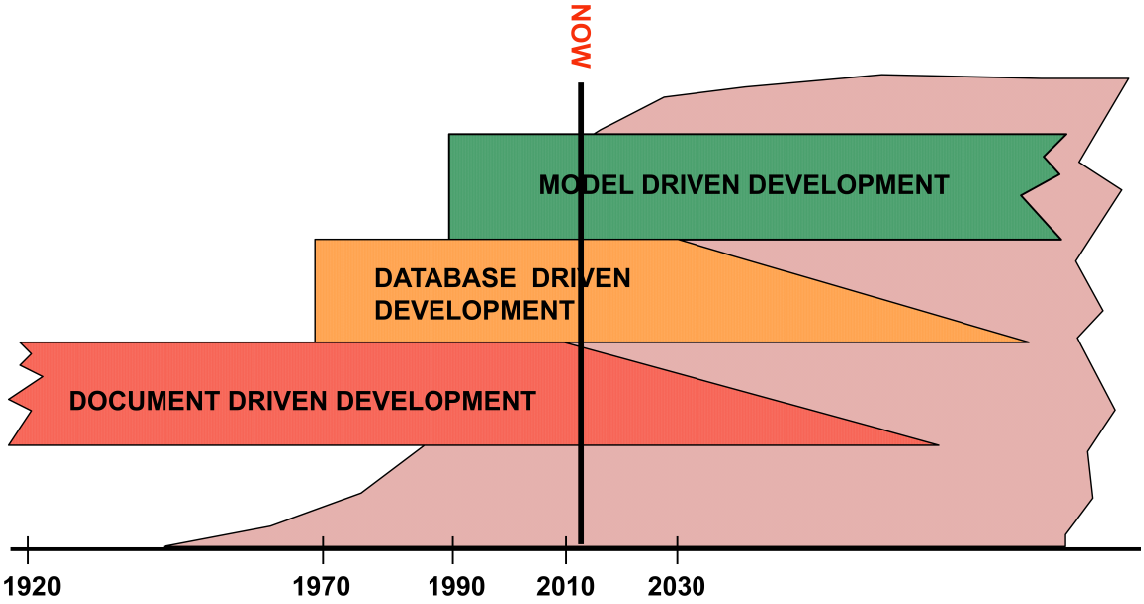


Figure 9 The Path We Are On

At present, however, the results of the problem space analysis is most often communicated to the system development program and its teams in the form of paper specifications that are then used as the basis for many specialized engineers to develop solution space models operating as isolated computer applications. The specialized engineers (reliability, mass properties, structural dynamicists, and many others) must present their results in design reviews and be subjected to questions by analysts, system engineers, and managers and they either receive an approval or direction that an alternative course of action be considered. The integration and optimization process relative to the evolution of a preferred solution space model or design concept occurs

largely through human communication between many narrowly but deeply specialized persons who often have considerable difficulty reaching a common understanding.

Progress will continue to improve our preferred problem space model and computer application developers will continue to try to perfect an effective implementation of that model for isolated operation as well as for at least semi-automatic model driven development linkage to solution space models. This will likely require a widely respected agreement on data transference between models. This probably remains the principal problem in reaching success on model driven system development and it will require a tremendous amount of work in the tool companies to make it possible for programs full of mere human beings to master the evolution of product designs while maintaining control over the system development process to the degree that it can be managed well.

Clearly, we are headed for a reality where many of the systems that we have to create will be more complex than in the past and require the service of more specialized domains populated by people with a narrower knowledge span and this combination will force the employment of more people at the cross-product and cross-domain boundaries to accomplish integration and optimization. So, there is a bias for the population of system engineers to increase in the future. Further, those responsible for enterprises and programs will have to start thinking of themselves as systems yielding even more opportunities for system thinking. In the development of product systems there are actually three systems interacting in the form of the product system being developed, the program responsible for the development, and the multi-program enterprise within which the program exists.

The work that the increasing population of system engineers will have to deal with will be very difficult and therefore very stressful but at the same time very enjoyable and rewarding for those who do it well. We could, of course, continue developing paper documents that are passed on to the next process step but that pattern does involve a great deal of human communication and it could be argued that human communication is the worst interface on planet Earth. Ideally, we would follow practices that encourage good communication while reducing the need for it. We can reduce the degree of communication difficulty in an enterprise by selecting a single problem space model such as the UML-SysML UADF that can be used to populate the resultant model with the requirements that are extracted from the modeling artifacts and make those requirements available for unambiguous application to the solution space models employed by the many specialists involved in evolving a design concept, preliminary design, and procurement plans.

Figure 10 illustrates the goal we appear to be moving toward in terms of development machinery. System engineers will apply a single universal problem space model to define the essential characteristics of viable solutions, hopefully, defining the problem such that multiple solutions are permissible and a best solution may be selected from those evaluated. The question is which mode will this machinery function in? Very likely every one will continue for some time in what we could call a manual mode that reflects what we do today.

In the manual mode every discipline has ways to use computer applications to develop and maintain their view of problem and solution spaces on a program. The related data is stored in computers networked together but are operated as islands with bridges between these islands formed only through human communication between the engineers owning those islands. Should it develop that the mass properties engineer finds it necessary to increase the mass of part of the

system due to the structural design solution, he/she will change the mass model. If the control system engineer happens to become aware of that change and concludes that it drives a change in a term in a control system equation and the corresponding software implementation, that change might be evaluated and a management decision reached to approve the mass and control system equation change. This kind of problem could easily go undetected. In this mode inconsistencies and errors are detected only by vigilance on the part of system engineers who understand the relationships at knowledge and product boundaries and clearly understand where those relationships exist in the system from good product interface models and an understanding of the specialists responsibilities relative to those product relationships.

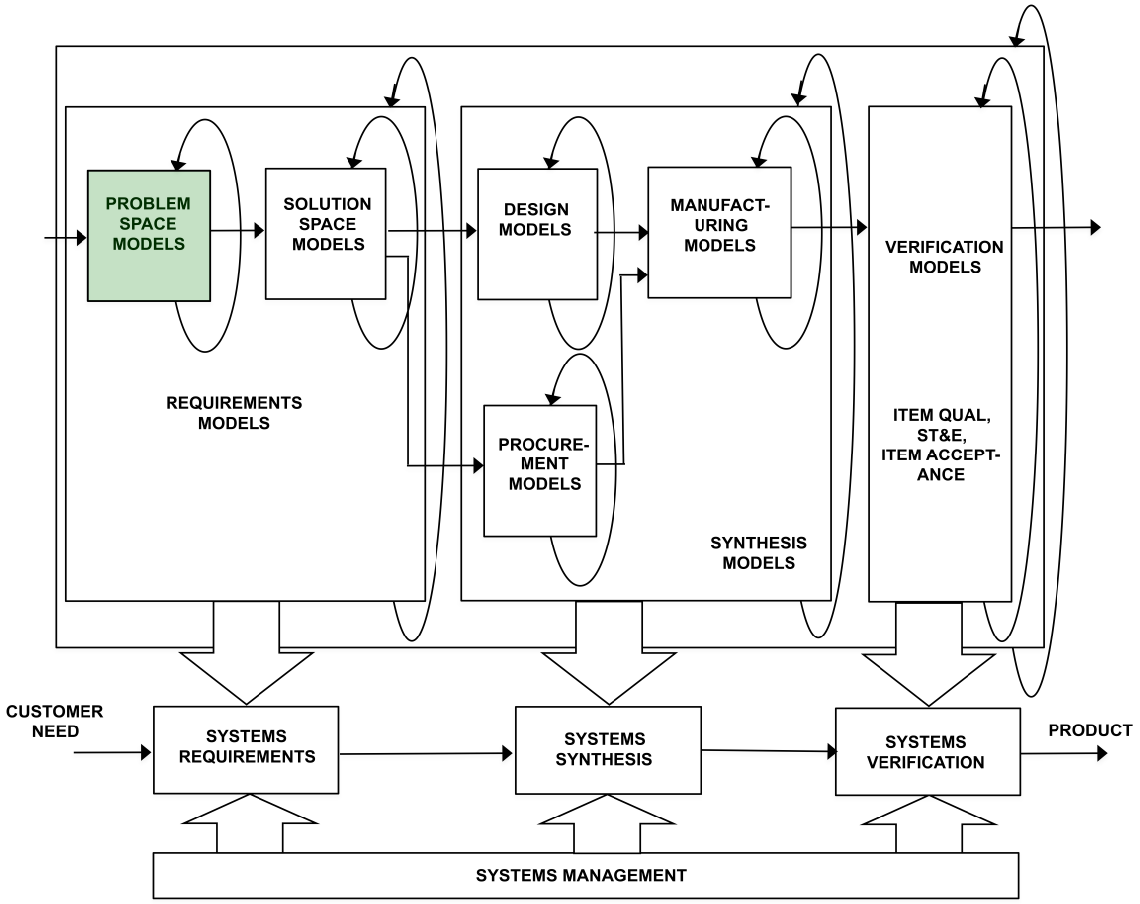


Figure 10 Model Driven Systems Engineering Environment

Over time, programs that gain good experiences from their model set in manual mode will experiment with a semi-automatic mode. Imagine for a moment that between the islands we had software operating so as to detect improper relationships between the content of pairs of island data sets. In this mode, that we can refer to as semi-automatic, a mass properties change is immediately detected, the relationships identified, and an inquiry made about its effect to the people responsible for the effected islands. In this case, the control system engineer hopefully will come forward, engage in the appropriate conversation, and a management decision reached to make the changes.

Now is the time for the test for just how daring the reader is. Suppose we let the connecting software not only detect errors but automatically correct them as well in an implementation of a fully automatic model driven development implementation that accepts the computer network as a fully integrated member of the team. It is possible that we will never operate a development program in full automatic mode but it is entertaining to think about how it might work out.

System engineers will have to understand the complete model driven development capability including how the problem space modeling results are coupled into the solution space models operated by the specialists, the capabilities of the solution space models, how the data is communicated across the model boundaries. And who and how the integration and optimization work will be implemented on the solution space side.

The results from the solution space models will then be applied to generate the design solution that will be coupled to the manufacturing machines and procurement controls. In some industries there is already a good connection between CAD and manufacturing while in others the connection is very weak. Procurement and the related logistics that causes procured product to be available on the production line on time in the right configuration will have to be very effective but that is already a fairly well polished art.

Those who are familiar with how completely fouled up this development process can become when the only means of coupling between development nodes is human communication will appreciate that reliance on machines to bridge these gaps will be extremely difficult to perfect but do not despair. It is common to believe that this will not take place successfully for another 15 years. But those who have been around this work for a while will recall how rapidly changes can take place once a clear path has been defined as in "..land a man on the Moon.."

It is likely that we will continue to apply a three step verification process involving qualification to prove that the product entities up the chain satisfy the requirements contained in the problem space models. Depending on the product line, some qualification articles might be manufactured essentially in accordance with regular production but in other product lines it may be necessary to manually manufacture them in an engineering shop as is so often done today. Item acceptance will be fully integrated into the production process. System test and evaluation will likely have to continue to involve operation in accordance with customer scenarios on programs dealing with weapons systems though there will be a more automated way to compare results with expected performance than we might see today.

4. Some Features of System Engineers We Might Wish to Preserve

System engineers serving on programs involving model driven system development will have to understand how to do their work, of course, but also understand why it is necessary. Ideally, these system engineers would also understand what the specialists do and generally how they go about it and what machinery they use. Much of the interaction work that a system engineer will have to accomplish with the specialists will have to be done on a computer workstation, no doubt, but one should guard against becoming so dependent on the machines that they avoid interacting with the human beings that do the work.

The author hopes that the work will not become so rigidly organized that human interaction is seldom possible. He recalls some very intense meetings where people were yelling and

screaming at each other because they felt so strongly about their position on a question. This work should be at least a little bit of fun. Will we close about our computers in worshipful obedience to the forms that we have to complete or retain a little spirit that may from time to time require a referee to settle disputes as suggested in Figure 11. T. Claude Ryan, President of Ryan Aeronautical Company, once told his team that was about to depart for Washington to pitch a new program, "If you realize you said something stupid I want you to immediately fall off the platform because I'd rather the customer thought you drunk than stupid." Certainly there is room for a good relationship between us humans and the machines that serve us along with some acceptance of a degree of enjoyable silliness that marks human interaction from time to time.

Figure 11 What Will Become of Us?

In general, there is a good future for system engineers with an ever increasing body of knowledge available to mankind and a continuing limitation on the capacity of the human mind for knowledge forcing a trend toward narrower and deeper specialists and a greater number of cross-domain boundaries combined with increasingly complex problems yielding to solutions involving systems with more difficult cross-product interfaces because it is at these cross-domain and cross-product boundaries that those responsible for the two sides of these boundaries have great difficulty in deriving sound decisions without the service of people called system engineers. Further, we appear to be resolving toward a single modeling method and there is a great deal of effort being directed at solving the remaining problems with model-driven system engineering that, when successful, could mean a substantial improvement in system development efficiency. But, there is reason for concern that we may allow the ideal balance between man and machine to be forced too far in the direction of the machine making our work life less enjoyable.

JOG SYSTEM ENGINEERING
GRAND SYSTEMS DEVELOPMENT TRAINING PROGRAM
PRESENTATION

FROM WHERE WE ARE TO WHERE WE ARE GOING IN SYSTEMS ENGINEERING

Who Is Jeff Grady?

CURRENT POSITION
- **Owner, JOG System Engineering**
- **System Engineering Consulting and Education Firm**

PRIOR EXPERIENCE
- **U.S. Marines**
- **General Precision, Librascope Division**
  - **Customer Training Instructor, SUBROC and ASROC ASW Systems**
- **Ryan Aeronautical Company (later Teledyne Ryan Aeronautical)**
  - **Field Engineer, AQM-34 Series Special Purpose Aircraft**
  - **Project Engineer, System Engineer, Unmanned Aircraft Systems**
- **General Dynamics, Convair Division**
  - **System Engineer, Cruise Missile, Advanced Cruise Missile**
- **General Dynamics Space Systems Division**
  - **Engineering Manager, Systems Development Department**

FORMAL EDUCATION
- **SDSU, BA Math; UCSD, Systems Engineering Certificate;**
- **USC, MS Systems Management with Information Systems Certificate**

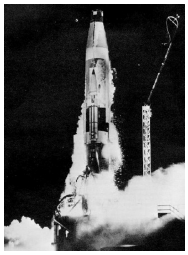
INCOSE **First Elected Secretary, Founder, Fellow, ESEP**

AUTHOR **System Requirements Analysis (2), System Integration, System Validation and Verification, System Engineering Planning and Enterprise Identity, System Engineering Deployment, System Verification, System Synthesis, System Management**

Systems Jeff Grady Worked On

USN/Librascope
ASROC/SUBROC
Computer Systems

USAF/GD Convair AQM 129
Advanced Cruise Missile



USAF/GD Atlas Missile

USAF/Ryan AQM-81 Firebolt

Ryan Aeronautical War Birds

USAF/Ryan Models 147G, NX, H, and J at Bien Hoa, SVN

USAF/Ryan AQM-34L Tom Cat
58 Combat Missions

U.S. Navy/Ryan
Model 147SK

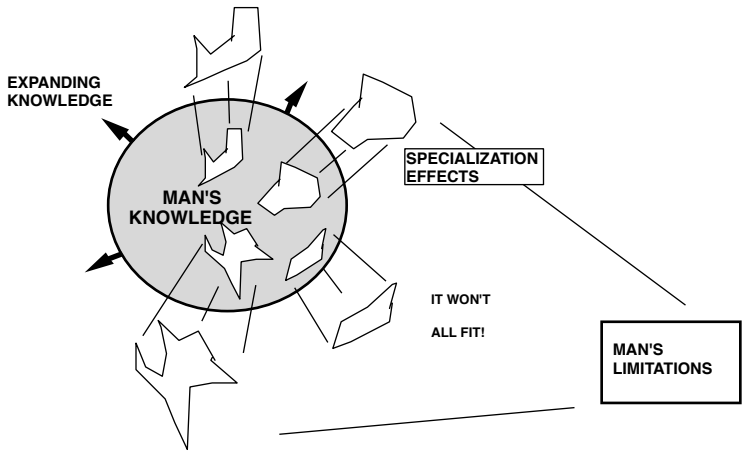
USAF/Ryan
BGM-34C

Outline

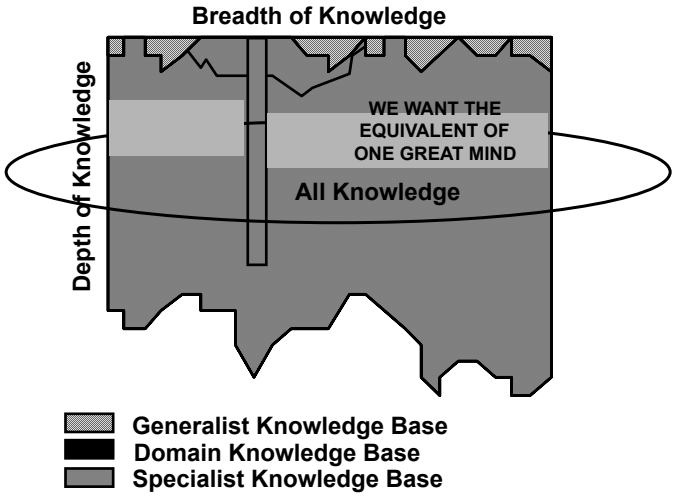
- **Where we are**
  - **A long term legacy**
  - **System Development in programs**
  - **Problem space modeling comes of age**
  - **Solution space modeling realities**
- **Where we are going**
  - **A future need for more system engineers**
  - **Model-based systems engineering – incoming**
  - **Remaining MBSE problems**
  - **But, what will become of the man-machine balance?**

Specialization of Knowledge

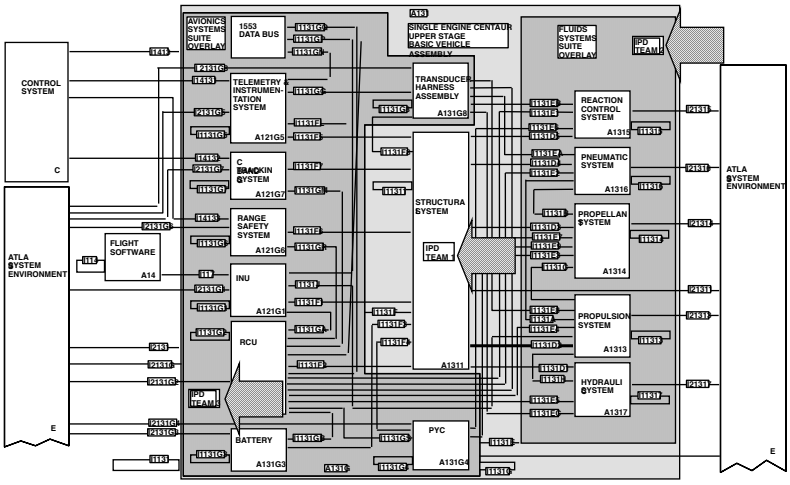
Knowledge Grows & We Have Our Limitations



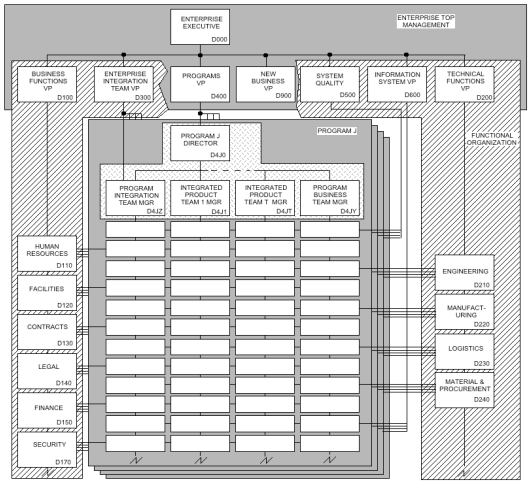
System Engineers Work at the Edges



Where System Engineers Live



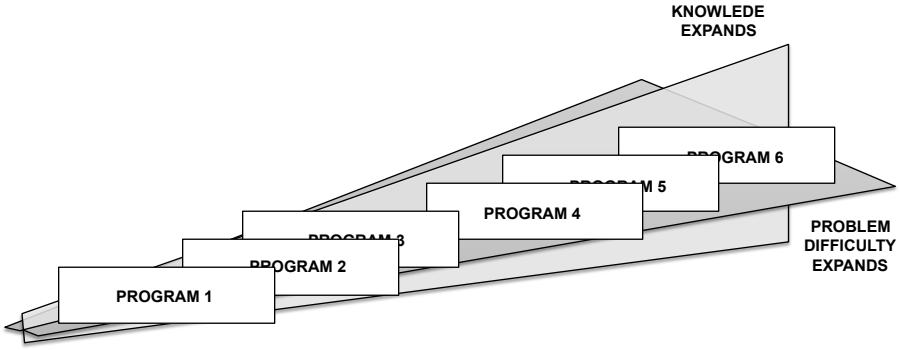
Program Organization Transform



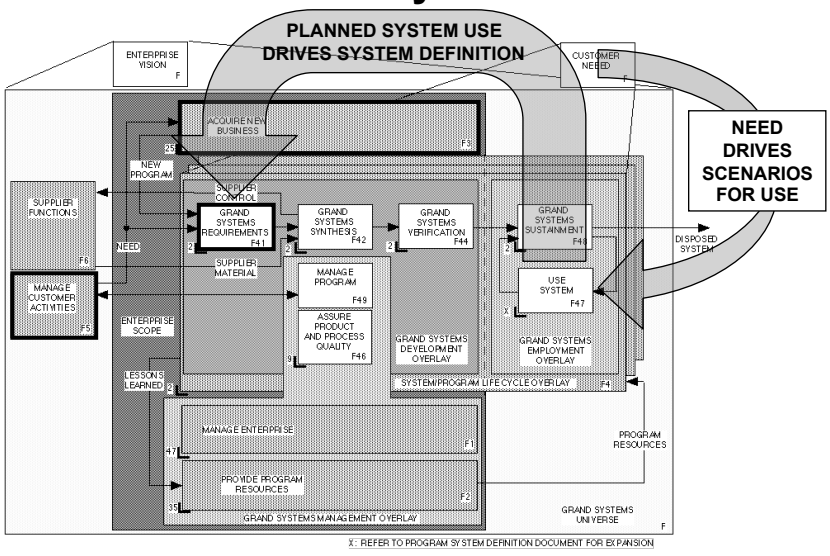
The System Development Sequence In Summary

- **Define the problem**
  - **Problem space models and solution space models**
  - **Specifications**
- **Solve the problem**
  - **Design, procurement/material, and manufacturing**
- **Prove it**
  - **Verification**
- **All within a sound technical management infrastructure**

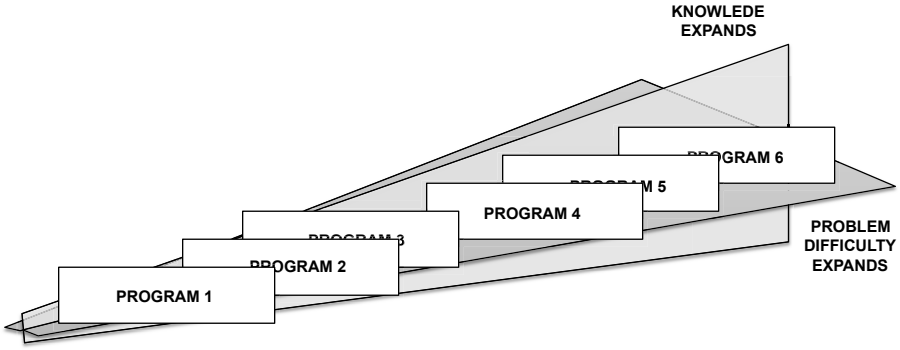
Programs In the Stream



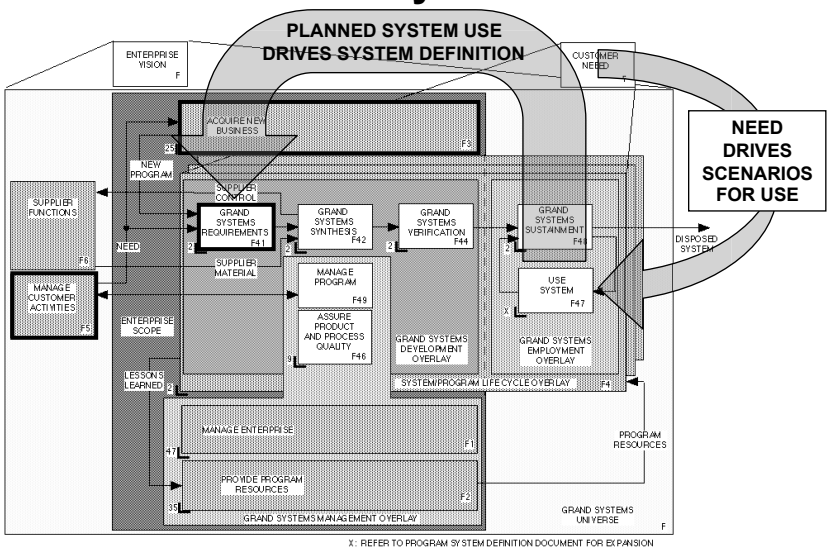
Common Life Cycle Model



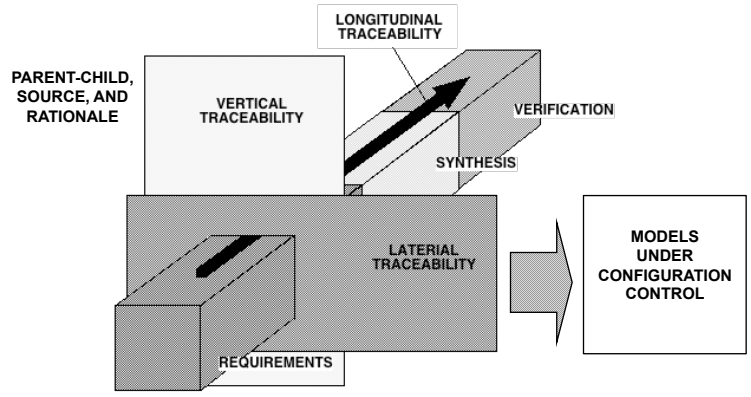
Programs In the Stream



Common Life Cycle Model

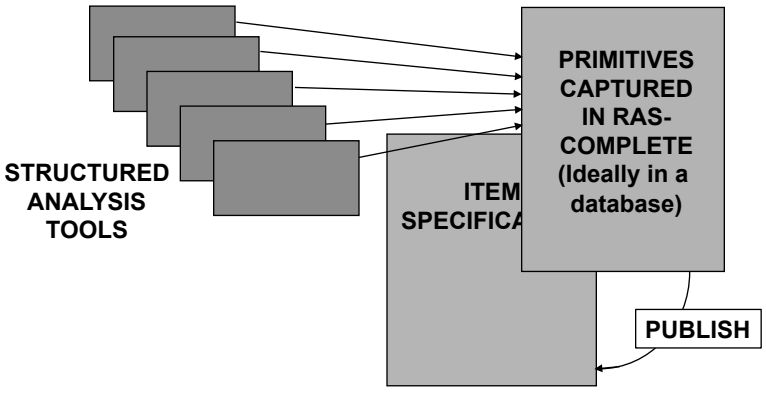


Maintain Three-Dimensional Traceability



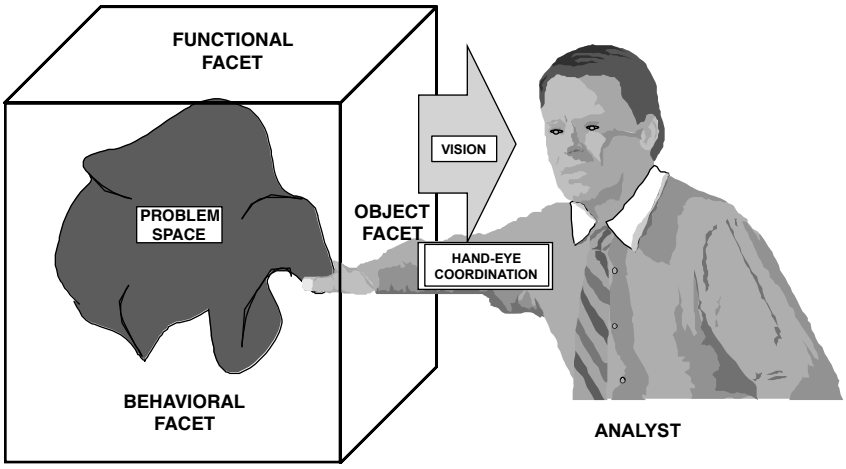
Lateral Traceability

Models as Characteristic List Builders

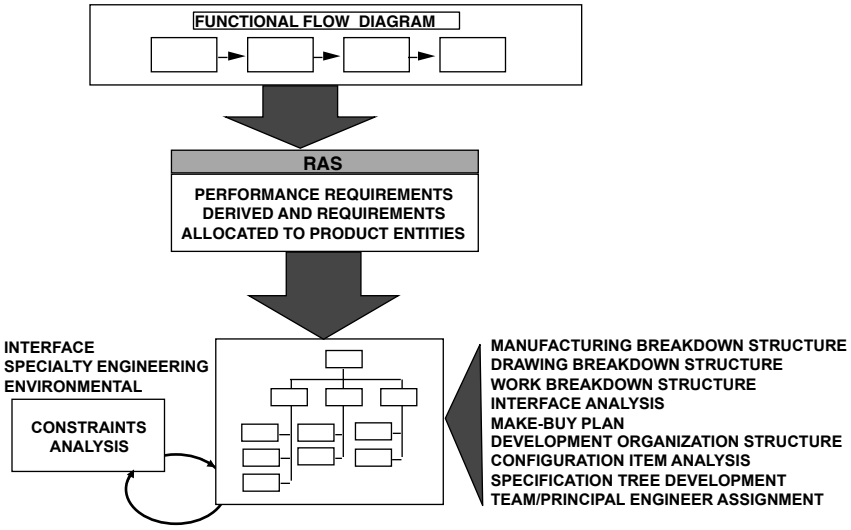


- It is not hard to write requirements.
- It is hard to know what to write them about and to determine what numbers to put in them

How Modeling Works

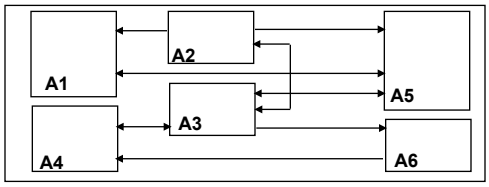


TSA UADF Function Allocation



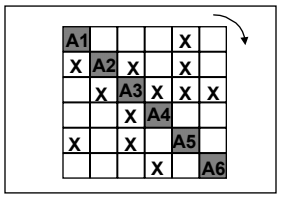
TSA Interface Definition Models

SCHEMATIC BLOCK DIAGRAMMING



- **Lines define interfaces**
- **Blocks are objects only from the product entity structure diagram**

N-SQUARE DIAGRAMMING



- **Marked intersections define interfaces**
- **Diagonal blocks are objects only from product entity block diagram**
- **Apparent ambiguity reflects directionality**

TSA Specialty Engineering Identification of Requirements

PRODUCT ENTITY STRUCTURE

SAR APPENDIX E

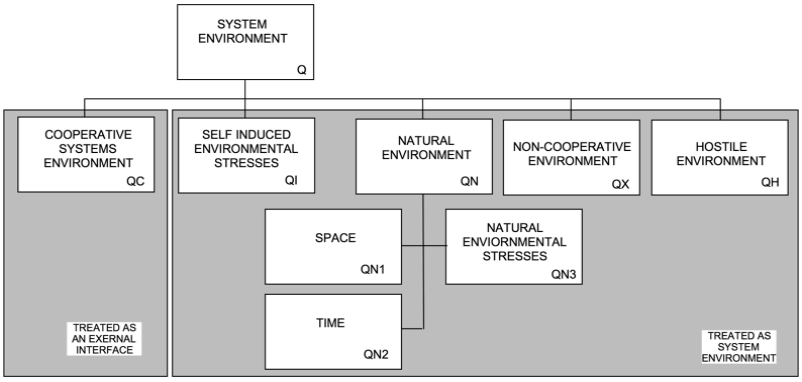
CONSTRAINTS	A11	A12	A13	A14	A15	A24	A25
H1	X	X		X			X
H2		X				X	
H3	X				X		
H4	X	X		X			
H5		X	X	X	X	X	X
H6			X				
H7	X	X	X			X	X
H8	X	X		X			X
H9			X	X	X	X	X
HA	X	X		X			X
HB		X		X		X	
HD	X	X	X	X			X
HD		X		X	X		

CONSTRAINT	ARCH
H7	A11
H7	A12
H7	A13
H7	A21

PRODUCT ENTITY-SPECIALTY ENGINEERING MATRIX (DESIGN CONSTRAINTS SCOPING MATRIX)

SPECIALTY ENGINEERING REQUIREMENTS FLOW INTO THE INDICATED SPECIFICATIONS THROUGH THE RAS

TSA Environment Subsets



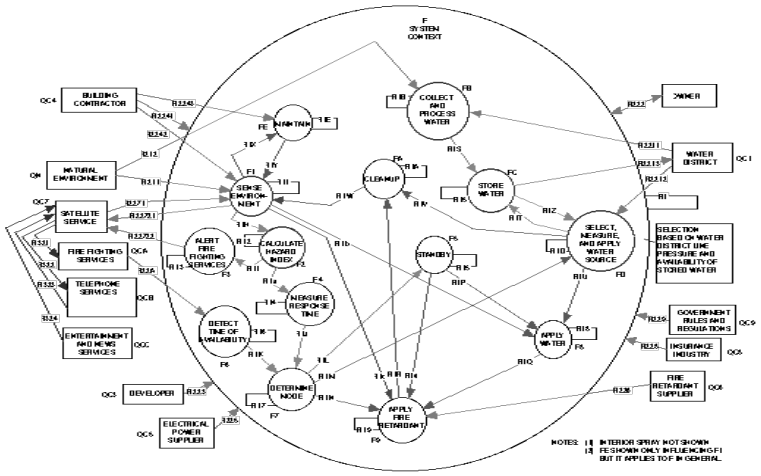
Some would add a software subset

Environmental Requirements Model

- **System**
  - **Identify spaces within which the system will have to function**
  - **Select standards covering those spaces**
  - **For each standard, select parameters that apply**
  - **Tailor the range of selected parameters**
- **End item**
  - **Build three dimensional model of end items, physical processes, and process environments**
  - **Extract item environments**
- **Component**
  - **Zone end item into spaces of common environmental characteristics**
  - **Map components to zones**
  - **Components inherit zone environmental requirements**

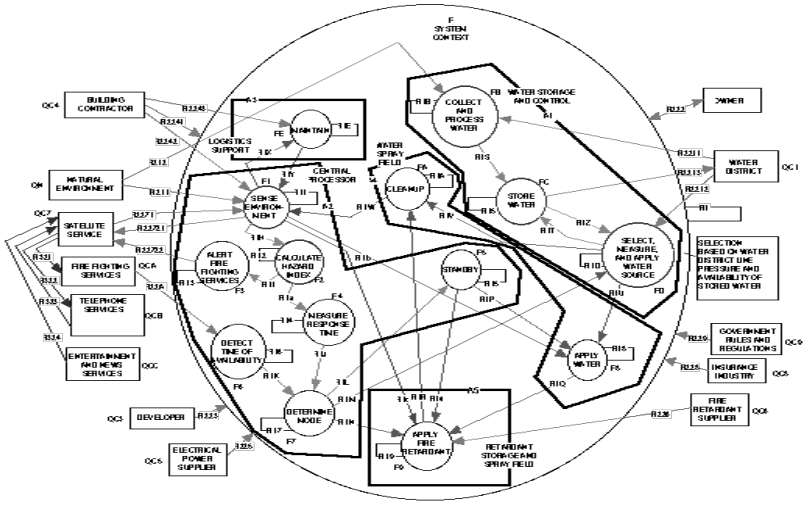
MSA/PSARE

Sample System Analysis – Context Diagram Expansion

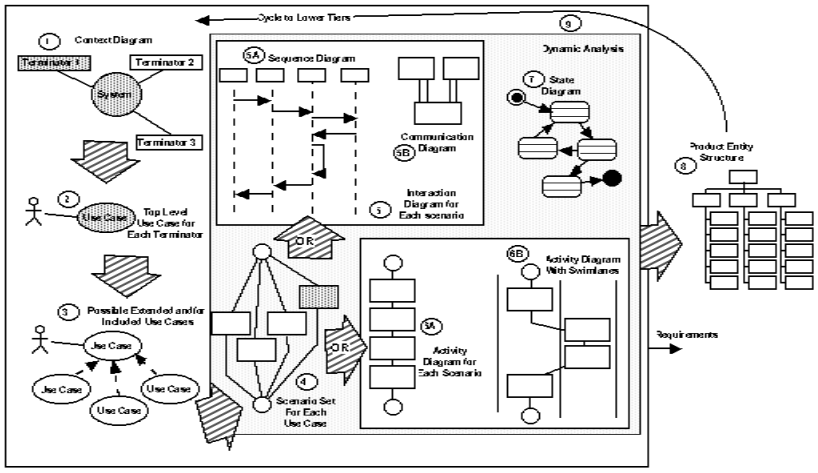


MSA/PSARE

Sample System Analysis - Super Bubbles



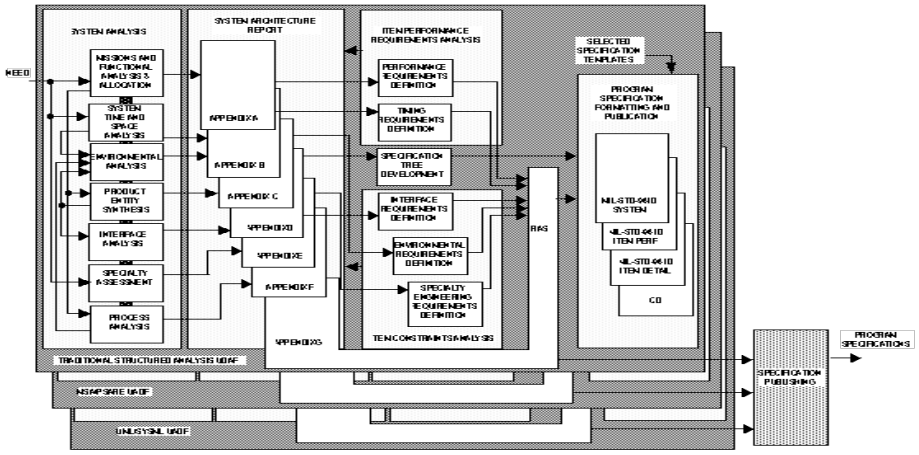
UML/SysML Dynamic Modeling Overview



Three Ways to Capture the Modeling

- **Within specification paragraph 3.1.3 on a program with few specifications**
- **In a system architecture report (SAR) referenced in paragraph 3.1.3**
- **Within the computer tool used to accomplish the modeling work with a reference in paragraph 3.1.3 to the tool content**
- **Within the modeling tools with a specification never published**

Lateral Traceability
Through the RAS and SAR Using TSA



Specification Template, Model Preference, and Responsibility Map

PARAGRAPH NUMBER	TITLE	RESPONSIBLE DEPARTMENT	PREFERRED MODEL	SAR APP
1	SCOPE			
2	APPLICALE DOCUMENTS			
3	REQUIREMENTS	D216-2	-	
3.1	Requirements Driven Sources	D216-2	-	
3.1.1	Non-Modeling Sources	D216-2	-	
3.1.1.1	Customer Need	D216-2	-	
3.1.1.2	Missions	D216-2	Mission Analysis	A
3.1.1.3	Threat	D216-2	Threat Analysis	B
3.1.1.4	Ad hoc Sources	D216-2	-	
3.1.2	Problem Space Modeling	D216-2	-	
3.1.2.1	Functional Flow Diagramming	D216-2	Functional Analysis	A
3.1.2.2	Functional Dictionary	D216-2	Functional Analysis	A
3.1.2.3	Requirements Analysis Sheet	D216-2	Functional Analysis	G
3.1.3	Solution Space Modeling	D216-2	Constraints Analysis	
3.1.3.1	Product Entity Modeling	D216-2	Product Entity Block Diagramming	C
3.1.3.2	Interface Modeling	D216-2	Schematic Block Diagramming	D
3.1.3.3	Specialty Engineering Modeling	D216-2		E
3.1.3.4	Environmental Spaces and Modeling	D216-2	Environmental Modeling	B
3.2	System Capabilities	D216-2	Functional Analysis	A
3.2.m	Capability m	D216-2	Functional Analysis	A
3.2.m.n	Performance Requirement n	D216-2	Performance Requirements Analysis	

Specification Template, Model Preference, and Responsibility Map

PARAGRAPH NUMBER	TITLE	RESPONSIBLE DEPARTMENT	PREFERRED MODEL	SAR APP
3.3	Interface Requirements	D216-2	Interface Requirements Analysis	D
3.3.1	Crossface Requirements	D216-2	Schematic Block Diagram	D
3.3.2	Innerface Requirements	D216-2	Schematic Block Diagram	D
3.3.3	Outerface Requirements	D216-2	Schematic Block Diagram	D
3.3.4	Government-Furnished Property (GFP) Interfaces	D216-2	N-Square Analysis	D
3.4	Specialty Engineering Requirements	D216-2	Specialty Engineering Modeling	E
3.4.1	Reliability	D216-4	Reliability Modeling	E
3.4.2	Maintainability	D216-4	Maintainability Modeling	E
3.4.3	Availability	D216-4	RAM Modeling	E
3.4.4	Deployability and Transportability	D231 D231	Logistics Analysis	E
3.4.5	Logistics	D231	Logistics Analysis	E
3.4.5.1	Maintenance	D216-4	Logistics Analysis	E
3.4.5.2	Interchangeability	D231	Logistics Analysis	
3.4.5.3	Supply	D231	Logistics Analysis	E
3.4.5.4	Facilities and Facility Equipment	D231	Logistics Analysis	E
3.4.5.5	Personnel	D231	Logistics Analysis	E
3.4.5.6	Training	D231	Logistics Analysis	E
3.4.6	Safety	D216-5	Safety Hazard Analysis.	E
3.4.7	Human Factors Engineering	D216-5	Human Engineering Analysis	E

Specification Template, Model Preference, and Responsibility Map

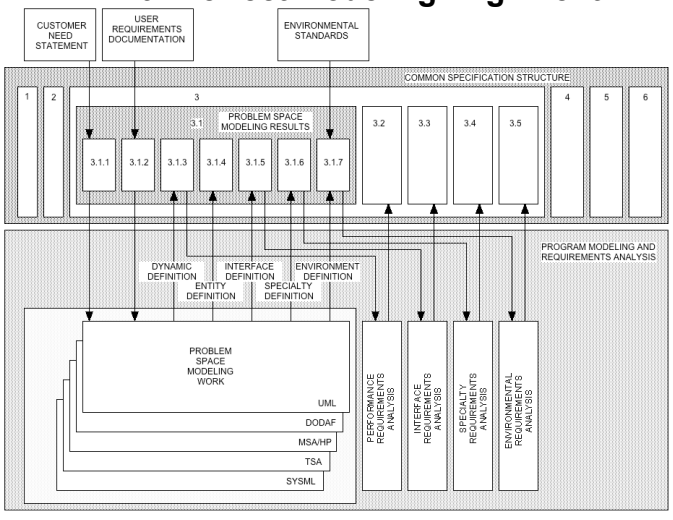
PARAGRAPH NUMBER	TITLE	RESPONSIBLE DEPARTMENT	PREFERRED MODEL	SAR APP
3.4.8	Security and Privacy	D216-6	System Security Analysis	E
3.4.9	Electromagnetic Radiation	D213-3	Electromagnetic Analysis	E
3.4.10	Lightning Protection			E
3.4.11	Producibility	D224	Manufacturing Requirements Analysis	E
3.4.12	Affordability			E
3.4.13	Computer Resource Requirements	D213-2		E
3.4.14	Design and Construction	D211-3	Configuration Management	E
3.4.14.1	Quality Enginering			E
3.4.14.2	Parts, Materials, and Processes	D216-7	Parts, Materials and Processes Analysis	E
3.4.14.3	Workmanship			E
3.4.14.4	Nameplates and Product Markings	D211-3	Configuration Management Techniques	E
3.4.14.5	Serialization			E
3.4.14.6	Mass Properties			E
3.4.14.7	Structural Properties			E
3.4.14.8	Shock and Vibration			E
3.4.14.9	Earthquake Survivability			E
3.4.14.10	Aerodynamics			E
3.4.14.11	Thermodynamics			E
3.4.14.12	Chemical, Electrical, and Mechanical Properties			E
3.4.14.13	Stability			E
3.4.14.14	Coatings			E

Specification Template, Model Preference, and Responsibility Map

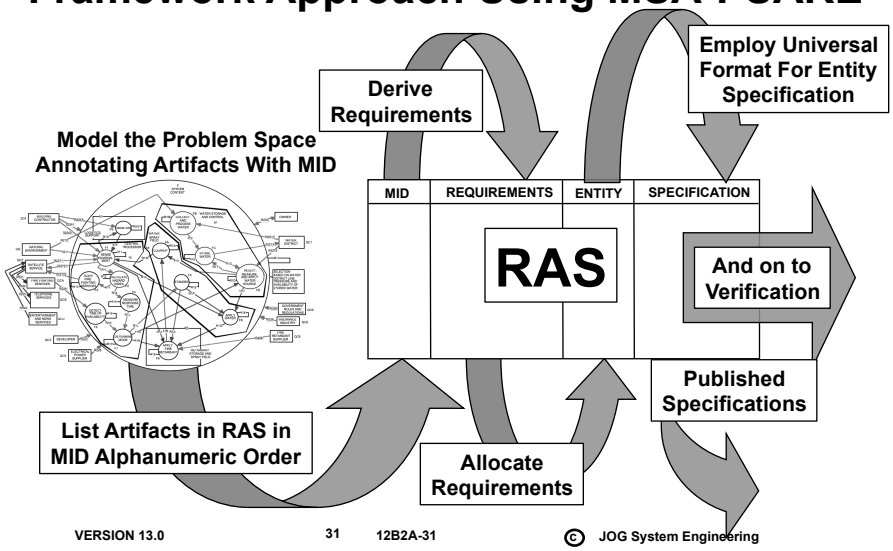
PARAGRAPH NUMBER	TITLE	RESPONSIBLE DEPARTMENT	PREFERRED MODEL	SAR APP
3.5	Environmental Requirements	D216-2	Environmental Require-ments Analysis	B
3.5.1	Natural Environmental Requirements	D216-2	Standards Analysis	B
3.5.2	Hostile Environmental Requirements	D216-2	Threat Analysis	B
3.5.3	Non-Cooperative Environmental Requirements	D216-2		B
3.5.4	Self-Induced Environmental Requirements	D216-2		B
3.5.5	Environmental Impact Limitations	D216-2		B
3.6	Precedence and Criticality of Requirements	D216-2		E
4	VERIFICATION			
5	PACKAGING			
6	NOTES			

Building Universal Specifications

With Perfect Modeling Alignment



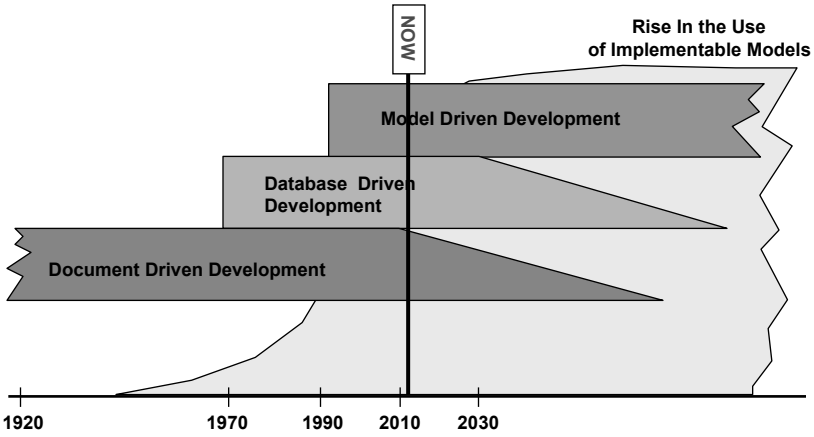
Universal Architecture Description Framework Approach Using MSA-PSARE



What Will the Future Look Like?

- A single model for the problem space - no matter how the specific product will be developed in hardware or software
- Requirements embedded in problem space models encouraging requirements compliance in design models with the specifications appearing in the form of models
- A connected series of models for design
- Inter-model effects observable directly rather than individual human interpretation of effects followed by conversation and action - can we do this?
- Verification linkage through models
- Eventual connection between the problem space modeling and CAD-CAM models.
- A business process model coordinated with engineering modeling

Development Evolution Timeline, Driving Methods Staging



05-15-2002 DATA UNSUBSTANTIATED

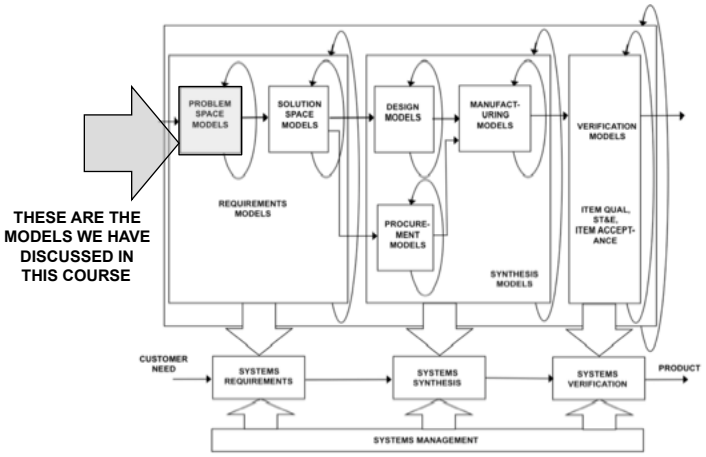
Action Items For You as a System Engineer

- Continue your studies of requirements work
- Come to an understanding about UML and SysML
- Within your company and programs develop modeling skills and work toward simplifying your combined set of models into a universal framework
- Work toward correlating the SW and HW development work patterns so as to encourage more effective integration
- Join INCOSE/NDIA working groups that deal with the issues covered in this paper and offer your ideas.

Model-Driven Challenges

- **Will it be possible for managers to avoid whiplash due to the speed of the analytical process?**
- **Can we provide adequate exposure of the on-going and dynamic modeling work to encourage sound management of the development process?**
- **Will it really be possible to build models that fully express the problem space essential characteristics (requirements) while permitting a solution space larger than a single solution?**

How Will the Links Be Closed?



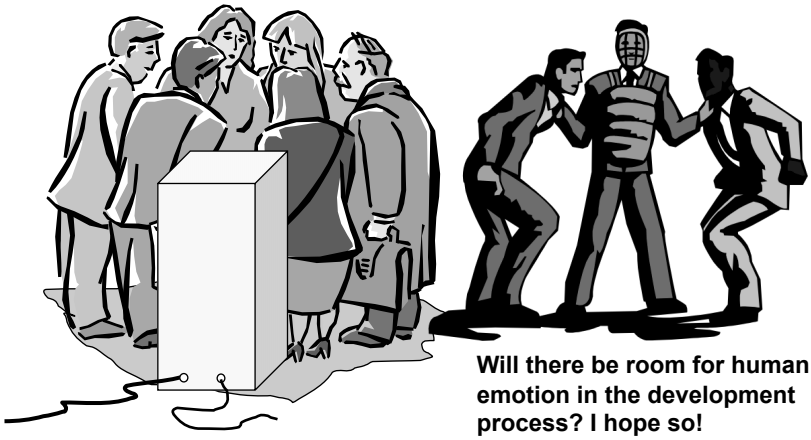
Model-Driven System Development Modes

- **Manual**
  - **Problem space modeling work is used as the basis for work accomplished by those working on solution space models**
- **Semi-automatic**
  - **Problem space modeling outputs applied to solution space models and humans reach conclusions to accept or adjust**
- **Full Automatic**
  - **Problem space results cause solution space response that is generally accepted and these results are applied to the procurement and design models and thereafter to the manufacturing machinery**

The Remaining Distance To Travel

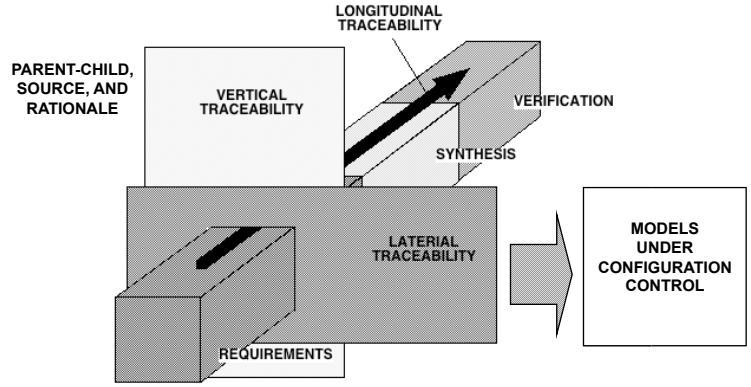
- **Close on one UADF**
- **Resolve the split between those who employ models to derive requirements and those who employ them to build "architectures" – one mode should be sufficient**
- **Perfect the tools supportive of UADF**
- **Solution space models**
- **Linkages between problem space and solution space models**
- **The machinery**
- **Ways to use the machinery**

How About Us Humans?



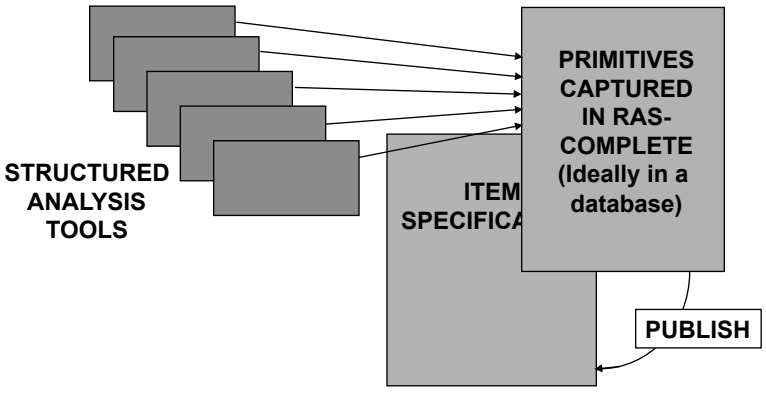
Will there be room for human emotion in the development process? I hope so!

Maintain Three-Dimensional Traceability



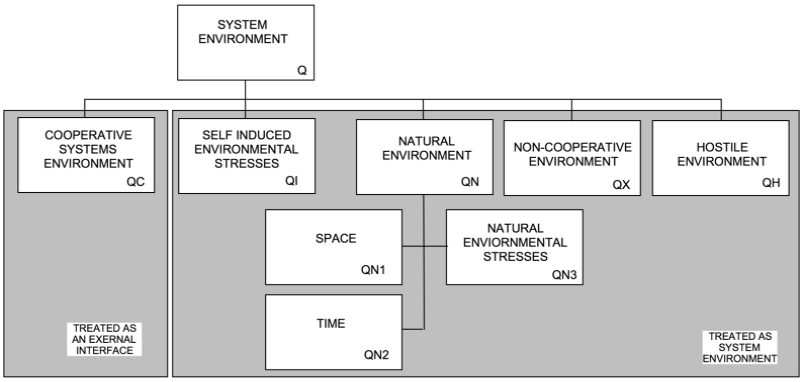
Lateral Traceability

Models as Characteristic List Builders



- It is not hard to write requirements.
- It is hard to know what to write them about and to determine what numbers to put in them

TSA Environment Subsets



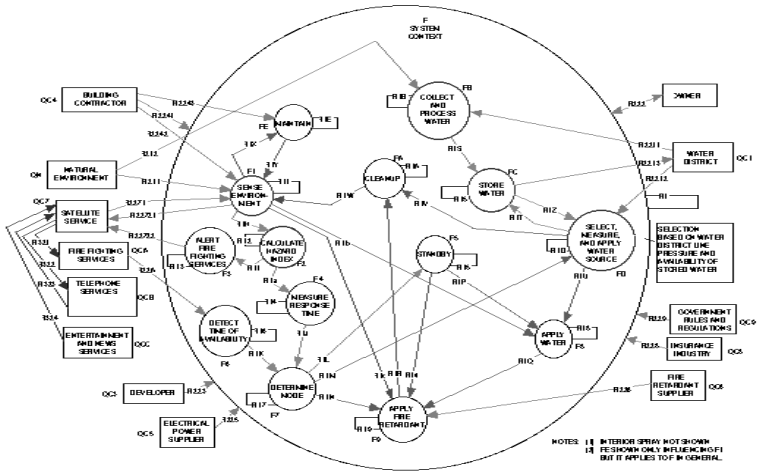
Some would add a software subset

Environmental Requirements Model

- **System**
  - **Identify spaces within which the system will have to function**
  - **Select standards covering those spaces**
  - **For each standard, select parameters that apply**
  - **Tailor the range of selected parameters**
- **End item**
  - **Build three dimensional model of end items, physical processes, and process environments**
  - **Extract item environments**
- **Component**
  - **Zone end item into spaces of common environmental characteristics**
  - **Map components to zones**
  - **Components inherit zone environmental requirements**

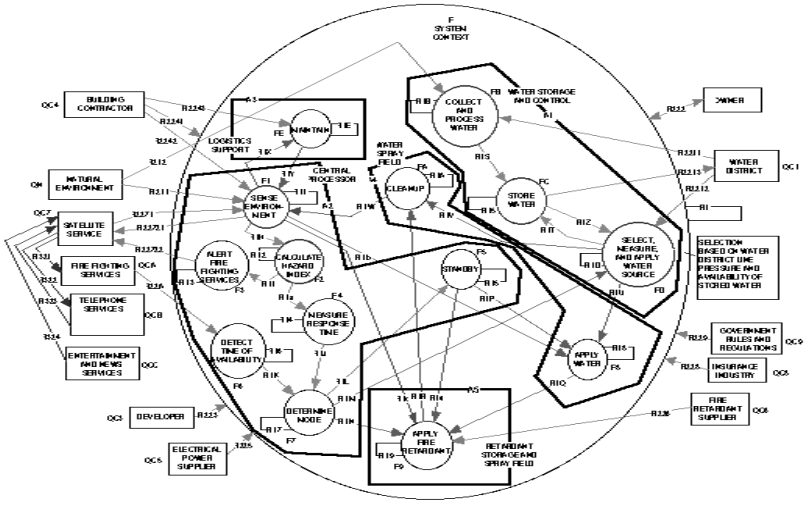
MSA/PSARE

Sample System Analysis – Context Diagram Expansion

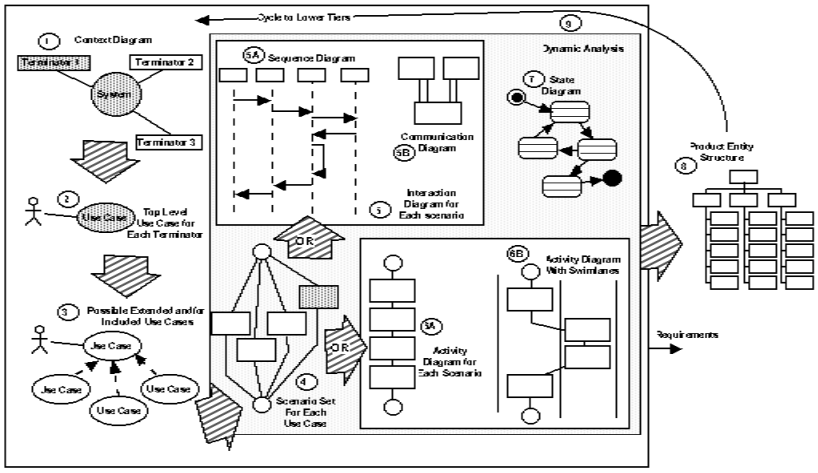


MSA/PSARE

Sample System Analysis - Super Bubbles



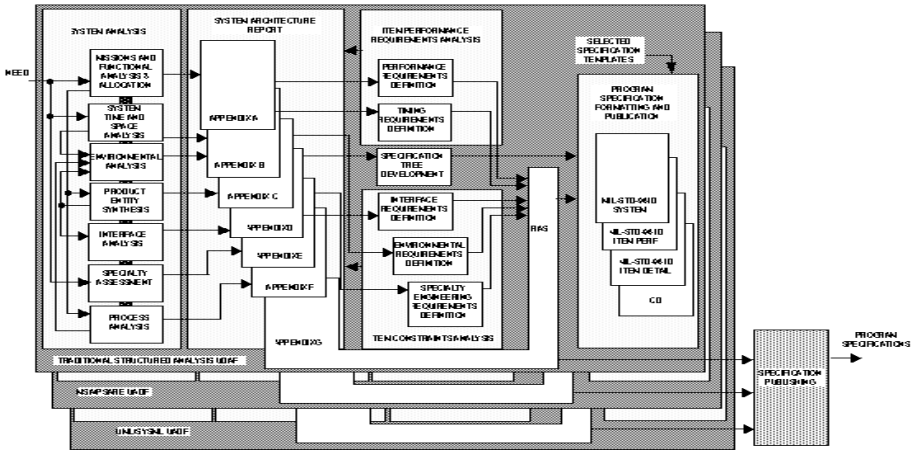
UML/SysML Dynamic Modeling Overview



Three Ways to Capture the Modeling

- **Within specification paragraph 3.1.3 on a program with few specifications**
- **In a system architecture report (SAR) referenced in paragraph 3.1.3**
- **Within the computer tool used to accomplish the modeling work with a reference in paragraph 3.1.3 to the tool content**
- **Within the modeling tools with a specification never published**

Lateral Traceability
Through the RAS and SAR Using TSA



Specification Template, Model Preference, and Responsibility Map

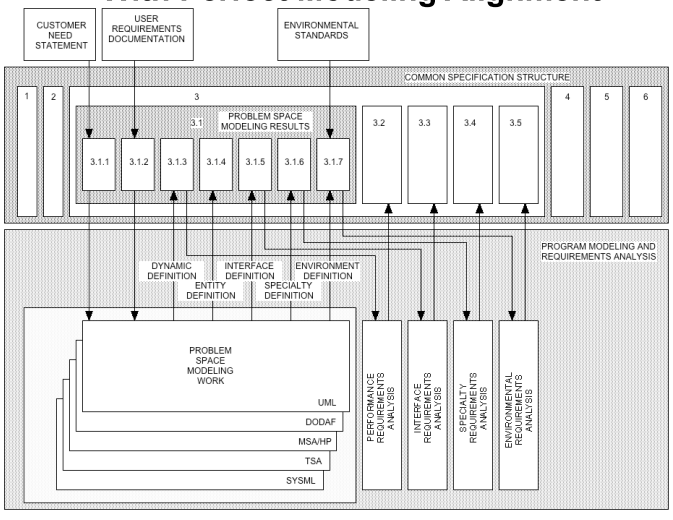
PARAGRAPH NUMBER	TITLE	RESPONSIBLE DEPARTMENT	PREFERRED MODEL	SAR APP
1	SCOPE			
2	APPLICALE DOCUMENTS			
3	REQUIREMENTS	D216-2	-	
3.1	Requirements Driven Sources	D216-2	-	
3.1.1	Non-Modeling Sources	D216-2	-	
3.1.1.1	Customer Need	D216-2	-	
3.1.1.2	Missions	D216-2	Mission Analysis	A
3.1.1.3	Threat	D216-2	Threat Analysis	B
3.1.1.4	Ad hoc Sources	D216-2	-	
3.1.2	Problem Space Modeling	D216-2	-	
3.1.2.1	Functional Flow Diagramming	D216-2	Functional Analysis	A
3.1.2.2	Functional Dictionary	D216-2	Functional Analysis	A
3.1.2.3	Requirements Analysis Sheet	D216-2	Functional Analysis	G
3.1.3	Solution Space Modeling	D216-2	Constraints Analysis	
3.1.3.1	Product Entity Modeling	D216-2	Product Entity Block Diagramming	C
3.1.3.2	Interface Modeling	D216-2	Schematic Block Diagramming	D
3.1.3.3	Specialty Engineering Modeling	D216-2		E
3.1.3.4	Environmental Spaces and Modeling	D216-2	Environmental Modeling	B
3.2	System Capabilities	D216-2	Functional Analysis	A
3.2.m	Capability m	D216-2	Functional Analysis	A
3.2.m.n	Performance Requirement n	D216-2	Performance Requirements Analysis	

Specification Template, Model Preference, and Responsibility Map

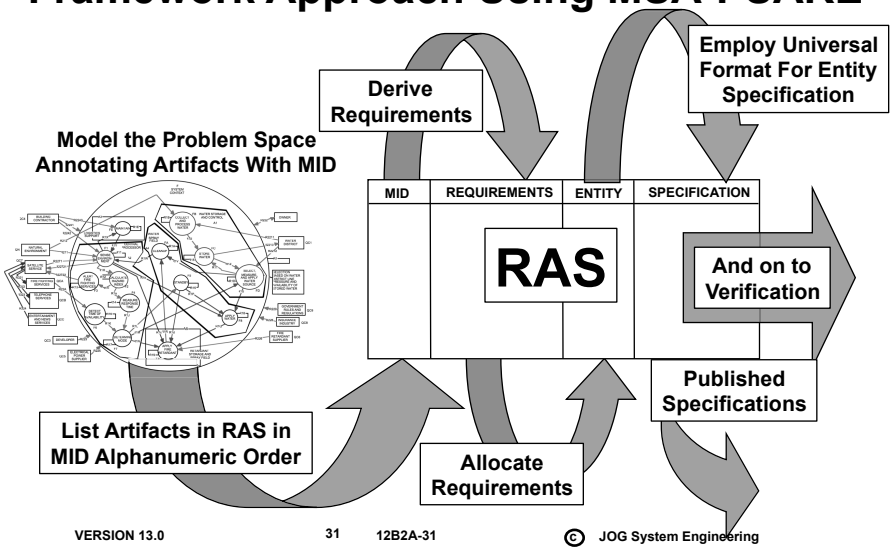
PARAGRAPH NUMBER	TITLE	RESPONSIBLE DEPARTMENT	PREFERRED MODEL	SAR APP
3.5	Environmental Requirements	D216-2	Environmental Require-ments Analysis	B
3.5.1	Natural Environmental Requirements	D216-2	Standards Analysis	B
3.5.2	Hostile Environmental Requirements	D216-2	Threat Analysis	B
3.5.3	Non-Cooperative Environmental Requirements	D216-2		B
3.5.4	Self-Induced Environmental Requirements	D216-2		B
3.5.5	Environmental Impact Limitations	D216-2		B
3.6	Precedence and Criticality of Requirements	D216-2		E
4	VERIFICATION			
5	PACKAGING			
6	NOTES			

Building Universal Specifications

With Perfect Modeling Alignment



Universal Architecture Description Framework Approach Using MSA-PSARE



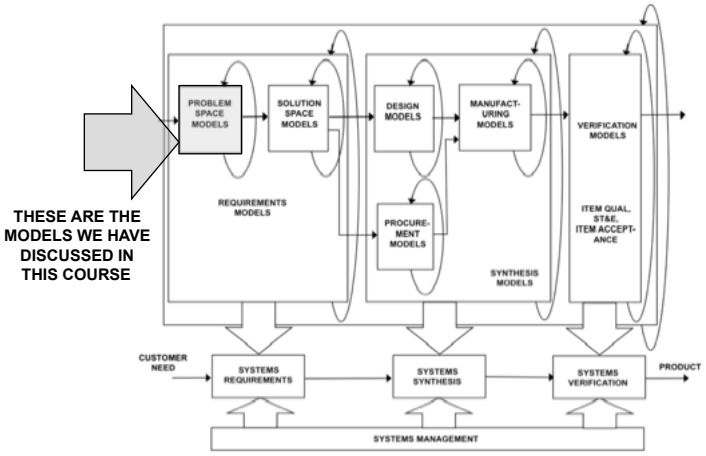
What Will the Future Look Like?

- A single model for the problem space - no matter how the specific product will be developed in hardware or software
- Requirements embedded in problem space models encouraging requirements compliance in design models with the specifications appearing in the form of models
- A connected series of models for design
- Inter-model effects observable directly rather than individual human interpretation of effects followed by conversation and action - can we do this?
- Verification linkage through models
- Eventual connection between the problem space modeling and CAD-CAM models.
- A business process model coordinated with engineering modeling

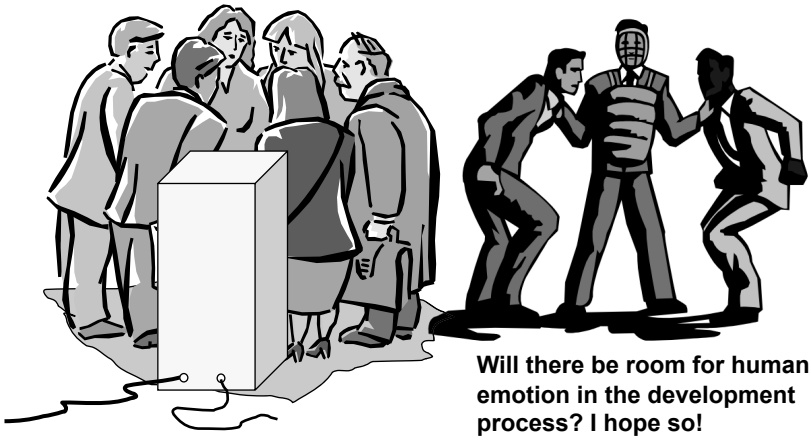
Model-Driven Challenges

- **Will it be possible for managers to avoid whiplash due to the speed of the analytical process?**
- **Can we provide adequate exposure of the on-going and dynamic modeling work to encourage sound management of the development process?**
- **Will it really be possible to build models that fully express the problem space essential characteristics (requirements) while permitting a solution space larger than a single solution?**

How Will the Links Be Closed?



How About Us Humans?



Will there be room for human emotion in the development process? I hope so!

