SAE TECHNICAL
PAPER SERIES

973217

The Tower of Babel: Language and Meaning in System Engineering

William D. Schindel
International Centers for Telecommunication Technology, Inc.

SAE The Engineering Society For Advancing Mobility Land Sea Air and Space
INTERNATIONAL

International Truck & Bus
Meeting & Exposition
Cleveland, Ohio
November 17-19, 1997

400 Commonwealth Drive, Warrendale, PA 15096-0001 U.S.A. Tel: (412)776-4841 Fax:(412)776-5760

973217

The Tower of Babel:Language and Meaning in System Engineering

William D. Schindel
International Centers for Telecommunication Technology, Inc.

ABSTRACT

Engineering complex systems is challenged by problems of *language*, increasingly as systems become more complex. Prominent are (1) *integration of subsystems* using *communication networks*, and (2) *integration of work* of engineers, product planners, suppliers, and customers. Both "hard engineering" and "soft process" cases encounter some of the same language problems in different settings. This paper briefly notes aspects of the structure of language and meaning and how they relate to these system engineering contexts. A practical, model-based technique, the Protocol Information Model (PIM), is described, to integrate both "hard" engineered systems and "soft" work processes which use "shared communication channels".

INTRODUCTION

The story of the Tower of Babel, not usual to the System Engineering context, reminds us of a problem faced in other settings: a project ends short of its goals when the workers cannot communicate with each other well enough to accomplish their intended result. In the case of the Tower of Babel, the workers started out speaking in a common tongue, but were scattered when Divine intervention drove their languages apart.

In designing complex systems, we usually face enough challenges that our first concern is not the risk of such a heavenly intervention. However, in complex systems projects we have learned that challenges of language and meaning, often of our own creation, are among the greater practical risks to success. In a sense, we often manage to fragment our *own* languages.

Two common cases of this hazard are:

1. Machine Communication: Use of shared inter-component communication networks, possibly industry standards based
2. Human Communication: Integrating the efforts of multiple staff, across Engineering, Product Planning, and Service organizations, or across supplier and customer organizations.

If you have not encountered these problems in severe form, it is likely that you will in the future, because systems are becoming more pervasive and more complex, the problems are more severe as the systems are more complex, and more frequently we are applying integrating networks and integrated cross-company and multi-company communication processes and workflows.

Probably several of these problems are already familiar to you:

1. Machine data communication channels over which message sets flow which are increasingly complex and diverse, but seem to lack unifying organization, and which contain "dialects" understood only by separated camps of specialists.
2. Explosive growth in message dictionaries, not matched by common understanding of the meaning of information on the channel.
3. Increasing problems in trying to understand, describe, communicate about complex system products; similar challenges trying to design, implement, integrate, manufacture, install, configure, verify, use, operate, manage, account for, monitor, control, maintain security of, predict, diagnose, repair and maintain, evolve, integrate, or extend those system products.

OVERVIEW

The following sections review a model of the problem of shared meaning which underlies the communication problem. We note the sometimes overlooked or surprising fact that the application level semantics of communication interfaces mean that "interfaces are not local", which confuses some work assignments. A practical technique for addressing this challenge, the Protocol Information Model, is introduced, and viewed in the context of the hard engineering network and soft work flow planning and management. Finally, lest we focus too much on a single semantic paradigm, the problem of

robustness in a complex world is noted as a reason why diversity of language and models has practical value.

COMMUNICATION: THE PROBLEM OF MEANING

The nature of language has provided philosophers, cognitive scientists, logicians, and others with challenging problems for generations. Even after lifetimes of study, basic understanding of the nature and facility for language remains a subject of argument and continued work, with significant new arguments and claims even recently.

Because this problem is beyond the scope of this paper, let us use a simple system engineer's model of the problem for our discussion here. A wise man advised that all models are wrong, but some models are useful. Our goal here is a practical tool, so we follow the admonition of including just enough, but not too much, in this simple model: the Protocol Information Model (PIM).

Communicating about complex systems leads System Engineers to create models which describe those systems. In a related previous paper [1], we reviewed the use of combined *Class and Containment Hierarchy* models to deal with description of the static structure and dynamic behavior of the most complex systems, and some of the typical misunderstandings and risks inherent in everyday thinking and communicating about complex systems. We apply here the combined Class and Containment Hierarchy view described in the reference.

Suppose we are modeling a family of integrated vehicles and their subsystems. Figure 1 illustrates one type of view we might construct, showing a *containment hierarchy* of engine subsystem assemblies. Figure 2 illustrates a different type of view might construct, showing a *class hierarchy* of engine types. These two different types of hierarchies are the subject of everyday thinking and communicating, but are quite different in nature.

Most of the complex inter-component relationships in systems involve interactions such as electrical, mechanical, hydraulic, economic, legal, or other types of relationships. Figure 3 illustrates a model view constructed to show various relationships of this nature.

Figures 1, 2, and 3 are all models about the same family of systems, but illustrate very different "slices" or views. They all may be thought of as views of a single integrated model, organized as shown in Figure 4. This integrated model arranges class and containment hierarchies along two orthogonal axes, and places all the inter-component relationships other than class and containment along the third axis. Although conceptually simple, correct, and complete, the resulting integrated model is typically too complex to view directly, so we satisfy ourselves with viewing "slices" of it, as in Figures 1-3.

These are *semantic* models. They describe the *meaning* of various components and component attributes, in the sense of their *relationship with other components*. They describe meaning by embedding in context--by relational connections.

Figure 1: Containment Hierarchy

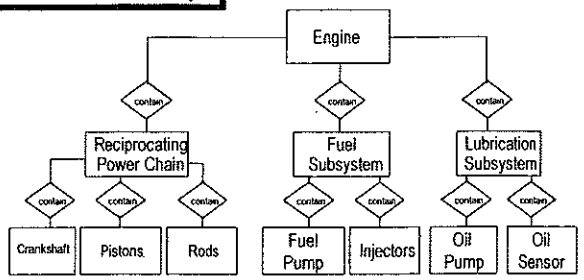


Figure 2: Class Hierarchy

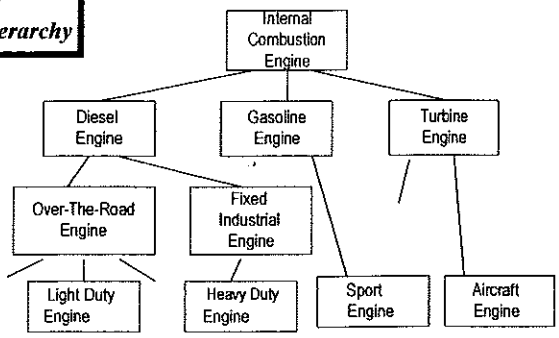


Figure 3: System Relationships

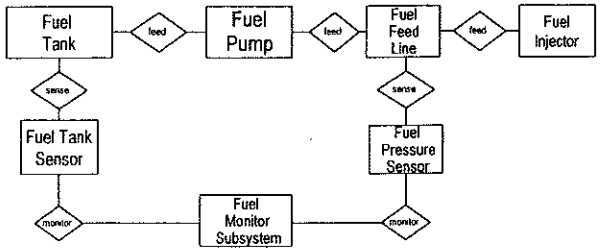
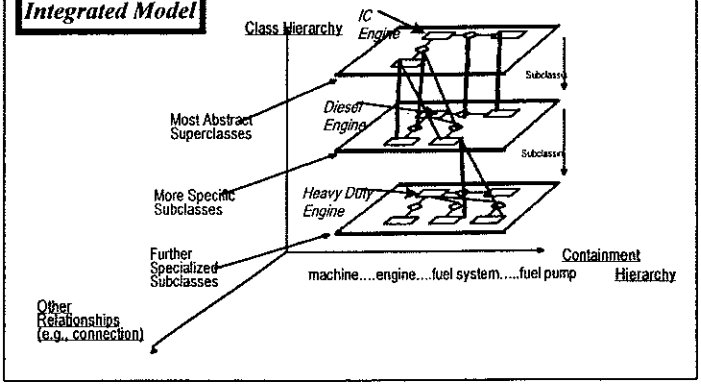


Figure 4: Integrated Model



COMMUNICATION: SEMANTICS VS. SYNTAX

When we communicate, whether between humans/work groups or between intelligent machine subsystems, that communication is *about* something. This "aboutness" is the realm of semantics, just discussed.

Two communicating entities also have shared rules about the form of possible messages they exchange to communicate. These rules may delineate the fields of a digital message, the grammar of an English sentence, or the possible composition of a songbird's melody. These rules are about *syntax*.

Syntactical rules tell what messages may be composed, and something about how to decompose, or parse, the message, into constituent parts. By contrast, semantic rules tell us how the constituent parts of the message may be interpreted, to give it meaning.

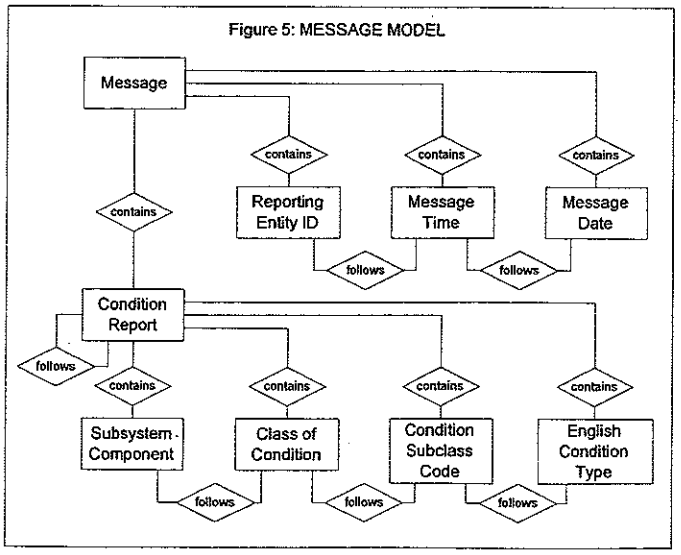
Consider the message:

"ENG 332179: FAULT 157: FUEL: Low Press 08:17 06-15-97"

This message can be parsed into major components:

1. The identifier telling us which Engine is reporting (possibly an engine serial number or ID code)
2. The FAULT message general type
3. The 157 indexed codification of the fault type
4. The subsystem component referred to (FUEL)
5. The English language version of the fault type
6. The time stamp
7. The date stamp

A slightly more complex message structure could be considered, by using a model of the message structure. We might allow that a single message can contain multiple fault or status items, with a single message time stamp, as in Figure 5:



Message models such as Figure 5 are *syntactical* models: they tell us about the form of possible messages. They are the rules for generating or decomposing messages. But, they do not describe what the message is *about*: its semantics.

However, we can add semantics to this model by showing explicitly the relationships of the message components to the model of the systems about which it is communicating, as in Figure 6.

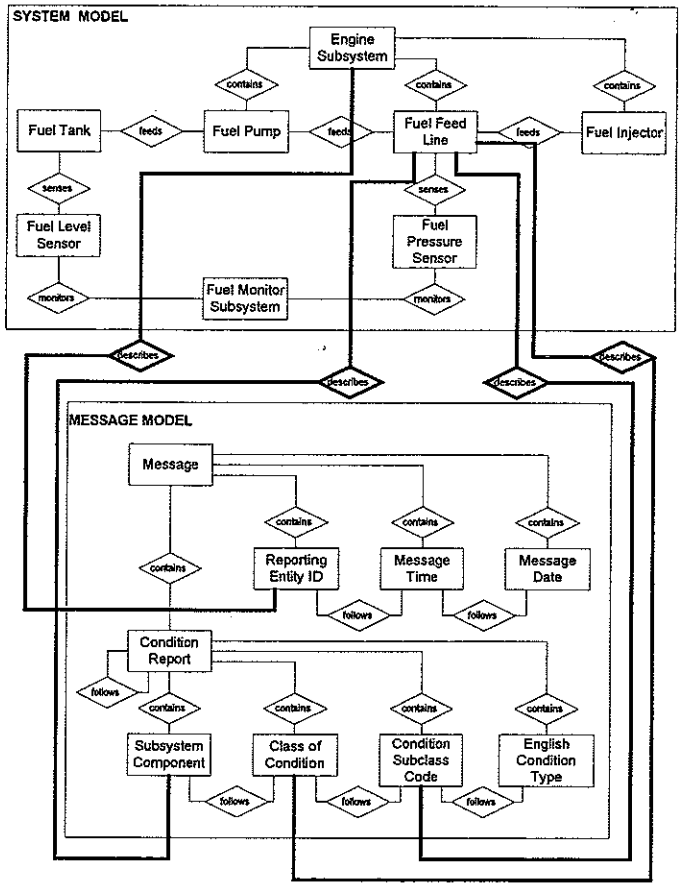


Figure 6: Integrated Syntactical and Semantic Model

Models such as Figure 6 describe both a system and how we are going to communicate about it, explicitly. They are the foundation of a Protocol Information Model (PIM), as discussed below.

WHY COMMUNICATION INTERFACES ARE NOT LOCAL

When we design data communication interfaces, the upper layers of their communication protocol stack are concerned with the above subjects. At the top-most layer, the Application Protocol is concerned with describing something about the system about which it "speaks". So, whether we make an explicit model such as Figure 6 or not, the Application layer

of the interface is deeply connected to the system it describes, as shown in Figure 6.

In real world projects, we often go to more trouble to model the syntax structure of the message than its semantics, assuming that "everyone knows what we mean" by the use of certain terms referring to the system. But when the subject system becomes really complex, this assumption becomes questionable.

This problem is compounded by the common misunderstanding that "interfaces are local"—that is, an electronic hardware view of the world that visualizes the interface as associated with the physical "spigot" local to some point of connection. For example, we may speak as if a communication interface is "local" to the physical cable connector that presents the communication circuit.

The problem with this view is that the application level of the interface—the part of the interface protocol stack that carries the real meaning of the data messages moving through the interface—is *about the whole system*, not some locally connected region of it.

This is another example of the peculiar occurrence of "the whole in the part" when we start to embed intelligence in systems: the ability of a subsystem to represent and manipulate information about another subsystem.

Figure 7 illustrates the idea by showing two subsystems that are communicating with each other about each other. The application level of the communication interface between two subsystems is really about the total content of the two subsystems—it is not "local" to the region of the physical interface. The complete specification of this interface, in the style of Figure 6, actually contains a model of the rest of the two subsystems, whether we draw it explicitly or not! The designers and specifiers of this interface should therefore include parties knowledgeable about the whole systems they are describing, not "just interface" specifications. The interface "component" *contains the rest of the system*!

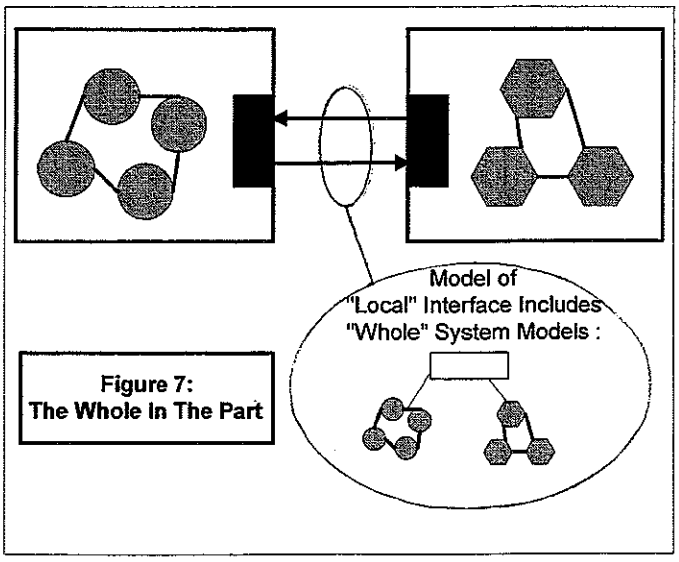


Figure 7:
The Whole In The Part

APPLICATIONS TO WORKING ORGANIZATIONS

The essence of the above situation arises because there is communication and because it is *about* a complex system. This is also the situation when we organize groups of people to do work together which is about a complex system. We often standardize aspects of the communication of this work, as in the use of standard forms, templates, document structures, etc. We often pay more attention to the syntactic description of these messages than their semantic models. The same problems arise, in a human context.

PROTOCOL INFORMATION MODELS: TOOLS FOR ADDRESSING THESE CHALLENGES

A *Protocol Information Model* (PIM) is a combined model that describes communication (appearing at a system interface, through a communication channel, or via a message) at both syntactic and semantic levels (and often also including lower levels of the protocol).

By including all these components in our models of communication, formally separated as needed, but formally related as well, we have an improved tool for dealing with increasing complexity in the systems we design, use, manufacture, and support.

The construction of a PIM is relatively simple in principle, although the work of building it will often surface underlying issues requiring effort to resolve:

1. Model the system about which communication will occur.
2. Model the messages which will carry the communication.
3. Model the relationships of (1) to (2).

UNIFICATION OF LANGUAGE VERSUS DIVERSITY: THE PROBLEM OF SITUATIONAL ROBUSTNESS

This approach, and the thrust of Reference [1], tell us how to create more unified views of systems, for thinking and communicating. Unification of language and description can have many economic benefits in simplifying rising complexity of systems. This is particularly true when we deal with configurable families of systems, as in product lines of integrated systems.

However, there is another edge to the Sword of Simplification in system engineering, and we return to the story of the Tower of Babel to recall it. As the story is told, the tower workers, dispersed by their fragmented language, scattered to different activities, ending the construction project. But, we are also told that the diversity that followed brought forth other benefits. What is the analogy in System Engineering?

If our engineering model is too unified, it risks losing Situational Robustness. As our systems encounter many different situations, we want them to continue to meet overall requirements which we hope to be able to describe. In real world complexity of complex systems, this may require multiply inherited superclass paradigms which cover a diverse

range of dissimilar perspectives. The techniques summarized here may be refined by this addition.

CONCLUSIONS

1. Increasing complexity of man-made systems challenges integration techniques in general, and communication interfaces in particular, as well as the effectiveness of communication by our own organizations.
2. Syntax and grammar are often overemphasized in place of semantics about the system.
3. Protocol Information Models (PIMs) provide a simple and explicit approach that addresses this problem in both the design of machine communication and organizational communication, about the systems.

REFERENCES

1. Schindel, William D., "System Engineering: An Overview Of Complexity's Impact", SAE Technical Paper Number 962177, October, 1996.
2. Hayakawa, S. I., *Language In Thought And Action*, fifth edition, Harcourt Brace Jovanovich, Publishers, Orlando, FL, 1990.
3. *Language, Thought, and Reality: The Selected Writings of Benjamin Lee Whorf*, J. B. Carroll, ed., MIT Press, Cambridge, MA, 1956.
4. Deacon, Terrence W., *The Symbolic Species: The Co-Evolution of Language and the Brain*, New York: W. W. Norton, 1997.
5. Shastri, Lokendra, *Semantic Networks: An Evidential Formalization and its Connectionist Realization*, Morgan Kaufmann Publishers, Los Altos, CA, 1988.
6. Pinker, Steven, *The Language Instinct: How the Mind Creates Language*, William Morrow and Co., New York, NY, 1994.
7. Language and Learning: The Debate Between Jean Piaget and Noam Chomsky, M. Piattelli-Palmarini, ed., Cambridge, Mass: Harvard U. Press, 1980.
8. Steven Cushing, *Fatal Words: Communication Clashes and Aircraft Crashes*, Chicago: U. of Chicago Press, 1994.
9. Chen, Peter, "The Entity-Relationship Model: Toward a Unified View of Data", *ACM Trans. On Database Systems* 1, 1(March), 9-36, 1976.
10. Booch, Grady, *Object-Oriented Analysis and Design With Applications*, Second Edition, Benjamin Cummings Publishers, New York, NY, 1994 .
11. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object Oriented Modeling and Design*, Prentice-Hall, New York, NY, 1991.
12. Jacobson, I.,Christerson, M., Jonsson, P., Overgaard, G., *Object-Oriented Software Engineering: A Use-Case Driven Approach*, Addison-Wesley Publishers, New York, NY, 1993.
13. Alexander, Christopher, *Notes on the Synthesis of Form*, Harvard U. Press, Cambridge, MA, 1964.
14. Shaw, Mary and Garlan, David, *Software Architecture: Perspectives On An Emerging Discipline*, Prentice-Hall Publishers, New York, 1996.

ABOUT THE AUTHOR

Bill Schindel has been involved in system design, development, support, and evolution for 28 years, including engineering, strategic systems marketing, and general management. He designed airborne military systems for the IBM Federal Systems Division, served on the faculty of Rose-Hulman Institute of Technology, and has founded and run two companies focused on complex systems—one in global telecommunications and one in general embedded systems markets. For the last ten years, he has been pursuing the application of models from biology to man-made systems. He is president of International Centers for Telecommunication Technology, Inc., Terre Haute, IN, and may be reached at schindel@ictt.com.