Draft #7 of Concepts Needed for System Static Structure

David W. Oliver

August 31, 2002

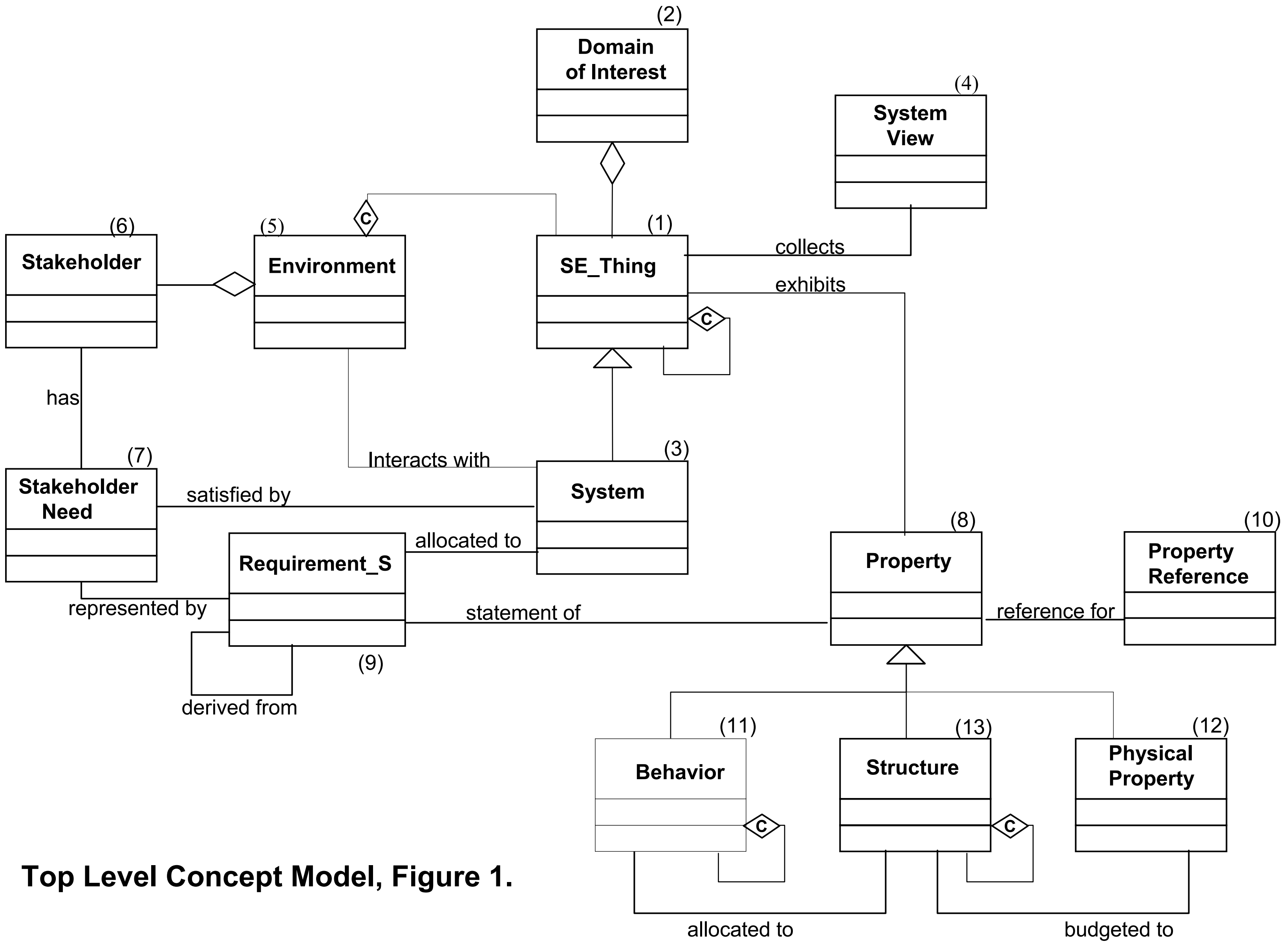
Wakefield, R.I.

Concept Model for Systems Engineering

Semantic Dictionary and Accompanying Model

The concept model consists of two interlocking parts. The first part is the semantic dictionary that defines each term. It is currently captured in an Excel spread sheet. The definitions have been written to satisfy the ISO standard for writing definitions that can be found on the BSCW web site. The definitions are an ordered set. Definitions lower in the set use terms found higher in the set. This helps prevent circular definition. Since the definitions are not arranged alphabetically, they are numbered with a reference number to aid in locating them.

Definitions according to the ISO standard define kinds of things, composition of things from other things, and associations among things. When one reads ten or more text definitions the usual mind finds it difficult to remember the many relationships implied. Hence a model accompanies the semantic dictionary. The dictionary explains meanings in natural language. The model captures the multitude of relationships in a graphic form so that the relationships can be scanned. The model is written in UML 1.X, with indications of semantics that are missing from the language.



Top Level Concept Model, Figure 1.

Top Level Model

The model needs to be read with reference to the definitions in the semantic dictionary. It starts with SE_Thing that is any thing on which repeated measurements can be made for the engineering purposes of interest. This is a necessary definition because otherwise it is not possible to verify that a design or implementation meets its requirements. SE_Thing is built from SE_Thing in a hierarchy. The aggregation symbol has a small "C" in it to show that what is meant is a decomposition into all of the parts. The special notation is used because this concept is missing form UML 1.X.

The Domain of Interest constitutes all the things of interest to the application.

System is a kind of SE_Thing and thus it is built of systems in a hierarchy and it must have measurable characteristics that are repeatable. What makes the System unique is that it has well defined relationships with all of the things with which it interacts. The collection of those things is its Environment. To have a system it is necessary to characterize what is in the system and what is in the environment along with the static and dynamic interactions between system and environment. The Environment contains SE_Things and Systems.

Different persons in engineering, manufacturing, maintenance, and management need different sets of information about the system. Manufacturing personnel need to know about all the materials, nuts and rivets that go into the system and how they assemble together. Maintenance personnel need to have diagnostic information and deal with replaceable units of the system. There are a very large number of such useful collections of information, each with its own context. System View provides for the collection of such sets of information, each set in a particular context.

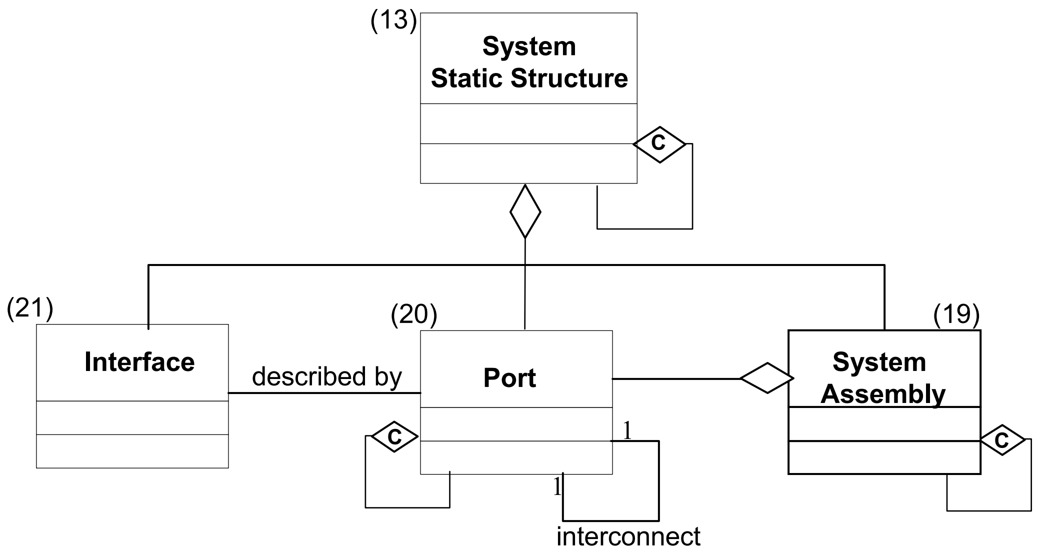
An important subset of things in the environment are the Stakeholders. These are all the persons and organizations with a need, preference, or interest in the system. Stakeholders may include manufacturer, owner, user of owner's services, user of the system, operator, maintainer, government regulator. Stakeholder Need represents their need, preference, interest, etc. in the system. If the System is designed and implemented well, then it satisfies these needs in a manner that is superior to competitive systems. It sells in the marketplace.

A Property is a named measurable or observable attribute, quality or characteristic of an se_thing. If you can measure it or observe it it is called a property. Properties have units, values, variances and probability distributions associated with them. They may be looked up in handbooks of properties of standard materials, they may be calculated from the structure of the thing, or they may be measured directly. In general they are tensors and may be a function of time. Because of the multiple ways of arriving at a property and its values, it is important to have a Property Reference that establishes the source of the information.

A Requirement is a statement of a Property that a System shall exhibit. The relationship to System is handled by allocating the requirement to the system that shall exhibit that property. This formality allows the engineer to consider alternative allocations to different systems that may fulfill the requirement. It is fundamental to trade-off among solutions. Requirements originate from Stakeholder Needs. As the design proceeds in levels of detail, requirements are derived from other requirements. These “derived from” relationships are preserved as traceability relationships. In a real world problem requirements will be changed from time to time. It is critical to trace from a requirement that has changed to other requirements impacted by that change.

It is useful to distinguish among three kinds of properties.

- Structure, the description of how a system decomposes into its parts and how the parts assemble to make the whole.
- Behavior, what the system does in response to the things in its environment. This includes both desired responses that satisfy needs, and prevention of undesired responses (failures) that can cause injury, destruction, or loss.
- Physical Property includes all the measurable or observable attributes, qualities or characteristic of an se_thing that cannot be observed in interaction with the environment. Additional instruments or tools are required to make the measurement or observation. Mass may require a scale for weighing, index of refraction may require use of an optical instrument.



System Static Structure, Figure 2.

System Static Structure is built from System Assembly, Port, and Interface. System Static structure decomposes hierarchically. This forces System Assembly and Port to also decompose hierarchically.

The System Assembly is simply a part or component list. The name used follows the STEP manufacturing point of view of looking at a part or component and talking about it as an assembly because their job is to assemble it. This is a place where it may be advisable for clarity to use the words component or part as an alias for System Assembly.

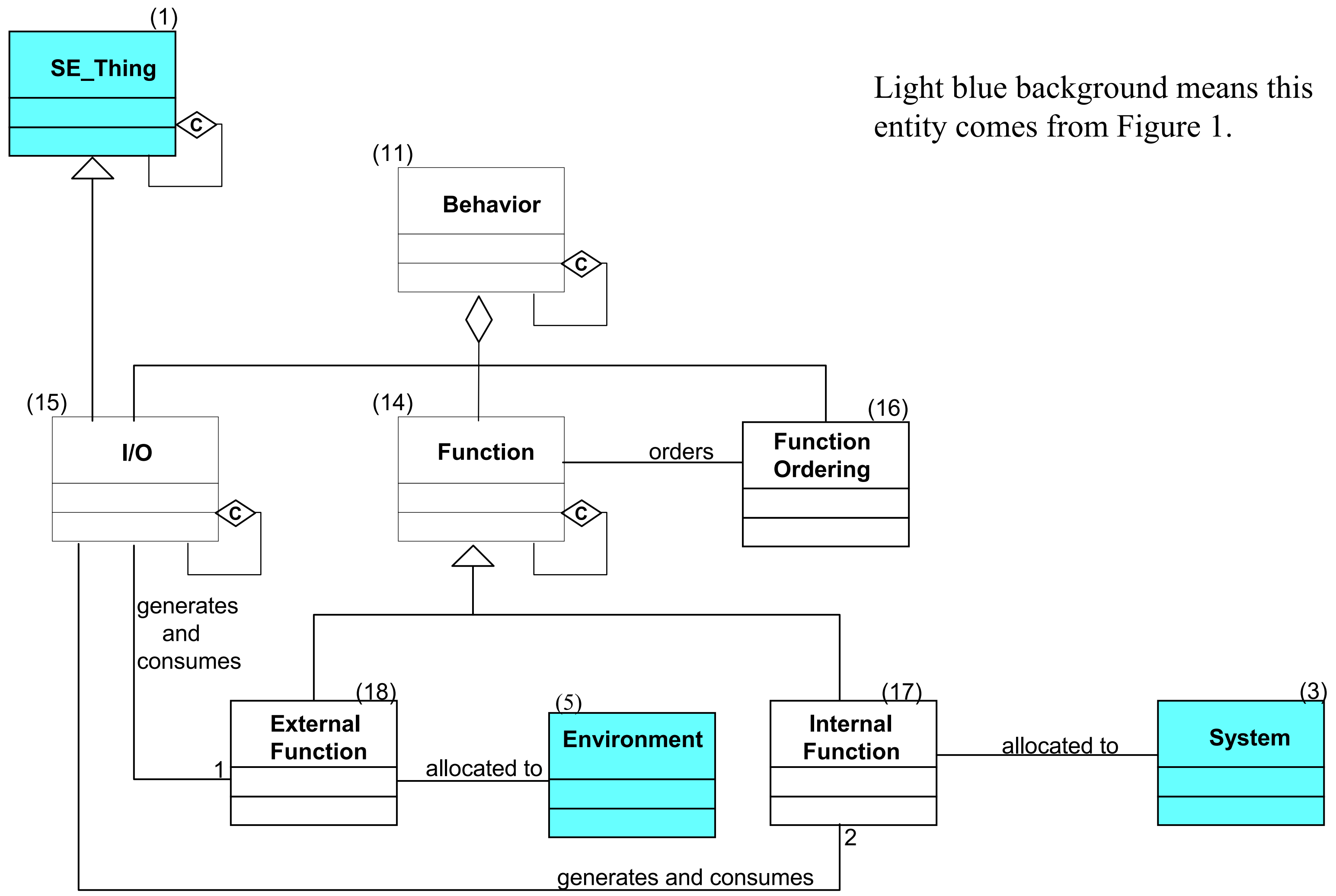
Each System Assembly (part or component) attaches to others at particular locations. These locations are called ports. This is a familiar idea when one thinks of the port on a power cord that plugs into a port on the wall to get electric power. It also applies to the surface of a bridge, a port, that interacts with wind, a port. In the second case the concept is less intuitive and more formal but it works. Ports connect to ports.

Each port has associated with it a description, an Interface, that describes the geometry, forces, transferred material or energy or information, protocols, how to assemble to it, and tests that may be required of the port-to-port connection. For two ports to be joined their interfaces must be compatible.

Behavior is built from Function, I/O (Input/Output), and Function Ordering as shown in Figure 3. Any SE_Thing may be I/O (Light blue shows an entity comes from Figure 1.). A Function is a entity of transformation that changes a set of inputs to a set of outputs. Function Ordering orders the functions such that it is possible to represent sequence, concurrency, branching, and iteration.

There are two major forms of representing Behavior. The continuous form emerged in systems engineering in the 1970's. It provides for completed functions to enable succeeding functions, for I/O to trigger functions, and for ordering operators to represent sequence, branching, and iteration. The SEDRES model represents this with a Petrie net model. UML 2.0 contributors may be using a Petrie Net model. If so, then these two models need careful comparison.

The discrete for of behavior representation emerged from automata theory

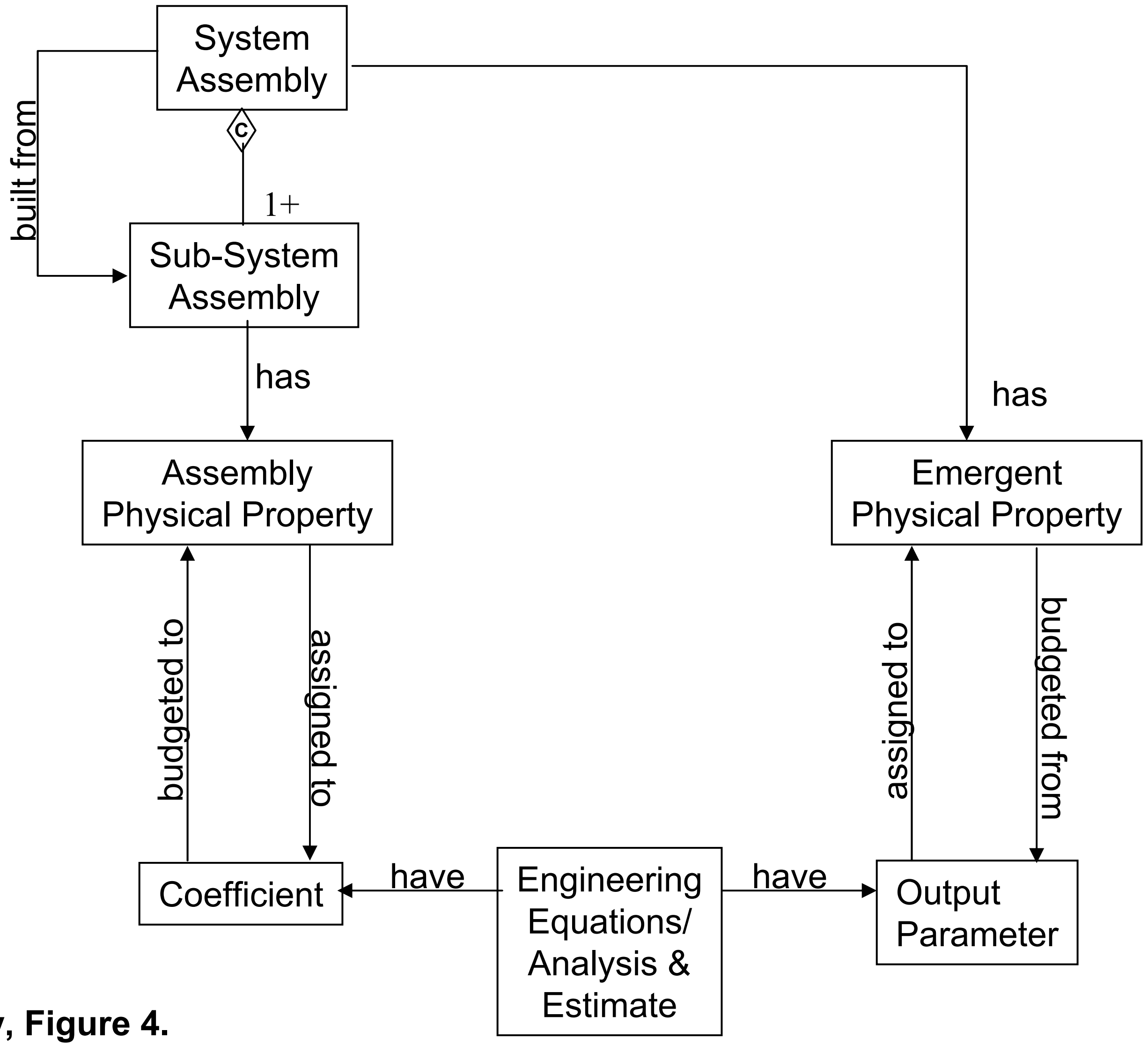


Light blue background means this entity comes from Figure 1.

Behavior, Figure 3.

and has matured into State Charts that provide for state explosion in highly concurrent models. SEDRES has a representation for this and has demonstrated model transfer between Statemate and Teamwork Real Time tools. In the UML community action Semantics are to provide a basis for state based behavior. These two approaches require careful correlation. The concept model here does not go beyond the very general notion of function ordering, but notes the critical importance of correlation among emerging detailed models.

Physical Property, its relationship to the Structure hierarchy and to analysis is shown in Figure 4. We expect this model to be improved in accuracy and in clarity. We are awaiting its completion and the replacement of Figure 4. by a member of the team. The key concept is that performance, behavior and physical properties result from the structure, the behavior and physical properties of the parts. They are not related to a class tree.



Physical Property, Figure 4.

Assemblies in the system assembly tree all have properties. Any
one assembly is an interconnection of assemblies one tier down in the tree.
The emergent properties of any assembly are a result of the properties,
interconnection,and interaction of the sub-assemblies from which it is built. The
relationships are very non-linear in the physical world. The components hydrogen,
oxygen, a spark assemble to make water and a great deal of energy with very
different characteristics than the lower level assemblies.

The basic relationships for emergent properties and budgeting of properties
are shown in Figure 4. A set of engineering equations or estimates are used
by systems engineers to budget properties to the interacting sub-assemblies
as a guide to designers at the lower level. When designs for all of the
sub-assemblies are available, their individual properties and interactions are
better defined. The same equations are used to calculate the emergent
properties of the complete assembly.

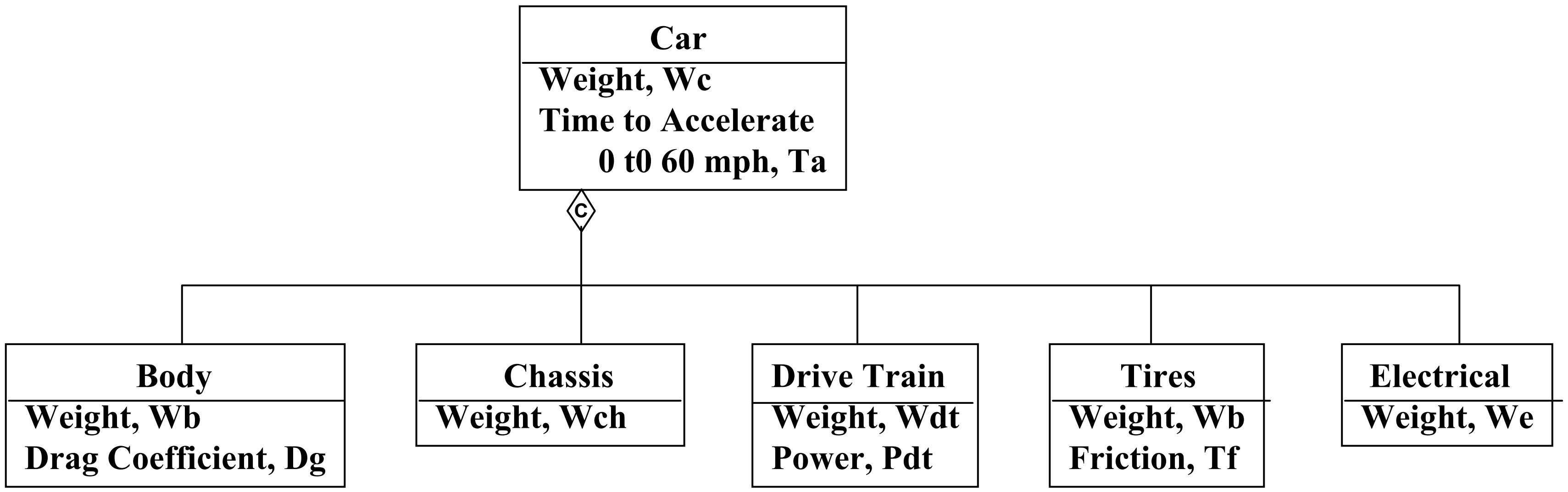
This practice is dictated by the nature of physical reality. It differs radically
from software engineering practices that emphasize classes with inheritance
and class trees. An example is described on the next slide and in Figure 5.

Emergent Properties and Budgeting of Properties Example

One may wish to develop a car that can accelerate from zero to sixty in 6.5 seconds or less. This is a required emergent property of the car. This behavior is a result of the power of the drive train, the air resistance of the body, the total mass of the car, and the friction of the tires on the road. These parameters are inter-related by a second order differential equation.

The differential equation is first used to budget target values of mass, power, drag coefficient, and tire friction to the appropriate components as targets for the designers. When the designs are available with definite property values, the same equations are used to calculate the emergent property, time for acceleration from zero to sixty mph for the car. See page 14 for a more technical view of the example.

Note that there may be several distinctly different approaches to the solution of what sub-components to use. Thus it is useful for the assembly to have attributes that indicate if it is an alternative or is selected as a solution, if it meets requirements, and what its regularization function value may be as the basis of selecting a particular solution from among the alternatives.



Engineering Equations

$Wc = Wb + Wch + Wdt + Wh + We$

$Wc * d^2\mathbf{x}/dt^2 + Dg * d\mathbf{x}/dt = Tf * Wc \qquad 0 < dx/dt < Pdt/(Tf*Wc)$

$Wc * d^2\mathbf{x}/dt^2 + Dg * d\mathbf{x}/dt = Pdt/(dx/dt) \qquad Pdt/(Tf*Wc) < dx/dt$

initial condition: x=o, dx/dt=0

compute: t, Ta, when dx/dt=60 mph

Figure 5. Ta is an emergent property of Car. Dg is an assembly property of Body

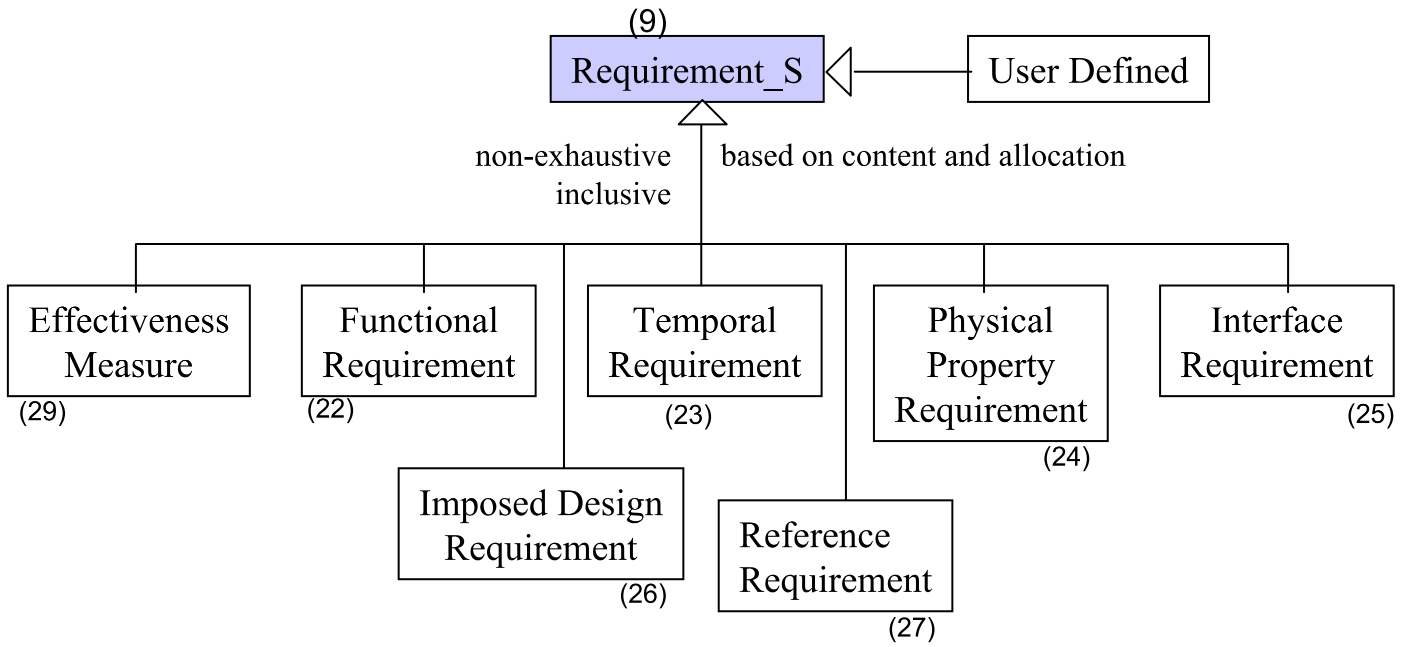
Allocation of Requirements

Depending upon their content, requirements are allocated to different parts of the information model. Requirements describing functions are allocated to functions, etc. This is a useful way of classifying requirements for the purpose of creating a logically consistent model or description of a system.

Within systems engineering there is no single standardized way of classifying requirements and many different classifications for different purposes are in use. The classification given in Figure 6. Is defined as shown because it is useful for the purpose allocating or assigning requirements.

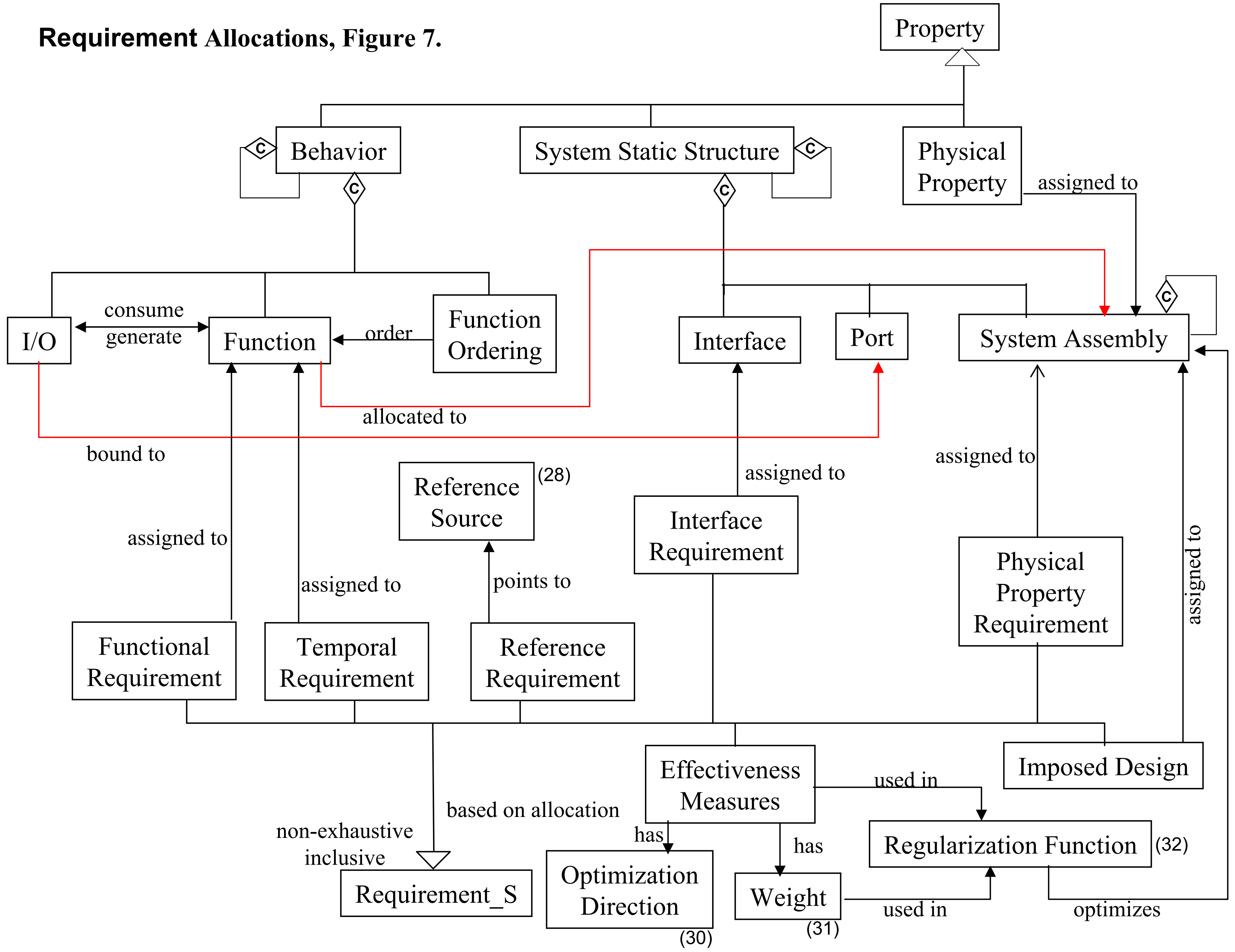
It is not possible to enforce any process with an information model and AP233 is intended to support both pest practices and other practices in use. Hence, any collection of requirements may contain compound requirements, contradictory requirements, and non-feasible requirements. Consequently the generalization/specialization of Figure 6. Is non-exhaustive and inclusive.

A Requirement Classification, needed to show how requirements are allocated to Behavior & System Structure



Classification of Requirements for the Purpose of Allocation, Figure 6.

Requirement Allocations, Figure 7.



Summary of Allocation Relationships

- Functional Requirements are assigned to to functions
- Temporal Requirements are assigned to functions
- Function is allocated to System Assembly (red used because of line crossings)
- I/O is bound to ports (red used because of line crossings)
- Interface Requirements are assigned to Interfaces
- Physical Property Requirements are assigned to System Assembly
- Imposed Design is assigned to the System Assembly on which it is imposed
- Reference Requirements point to a Reference Source that may contain requirements of all the kinds in the classification

Physical Property and Time

Figure 8. Shows draft models for Physical Property and Time. Physical Property and System Assembly are under study by a team member and improved models are expected for Figure 8. and Figure 4.

The model for time in Figure 8. is preliminary and needs discussion.

- Continuous Time is a dimension along with three spatial dimensions used by science and engineering to describe reality using math. It has no past, present or future.

- Present Time recognizes a standard of year, month, week, day, hour, minute, and second to represent past, present, and future. It is the basis of plans and schedules.

- Time Interval provides a time duration that may be assigned to a task or function to represent how long the task will take for completion.

- Start Time is a Present Time that states where in Present Time a Time Interval begins.

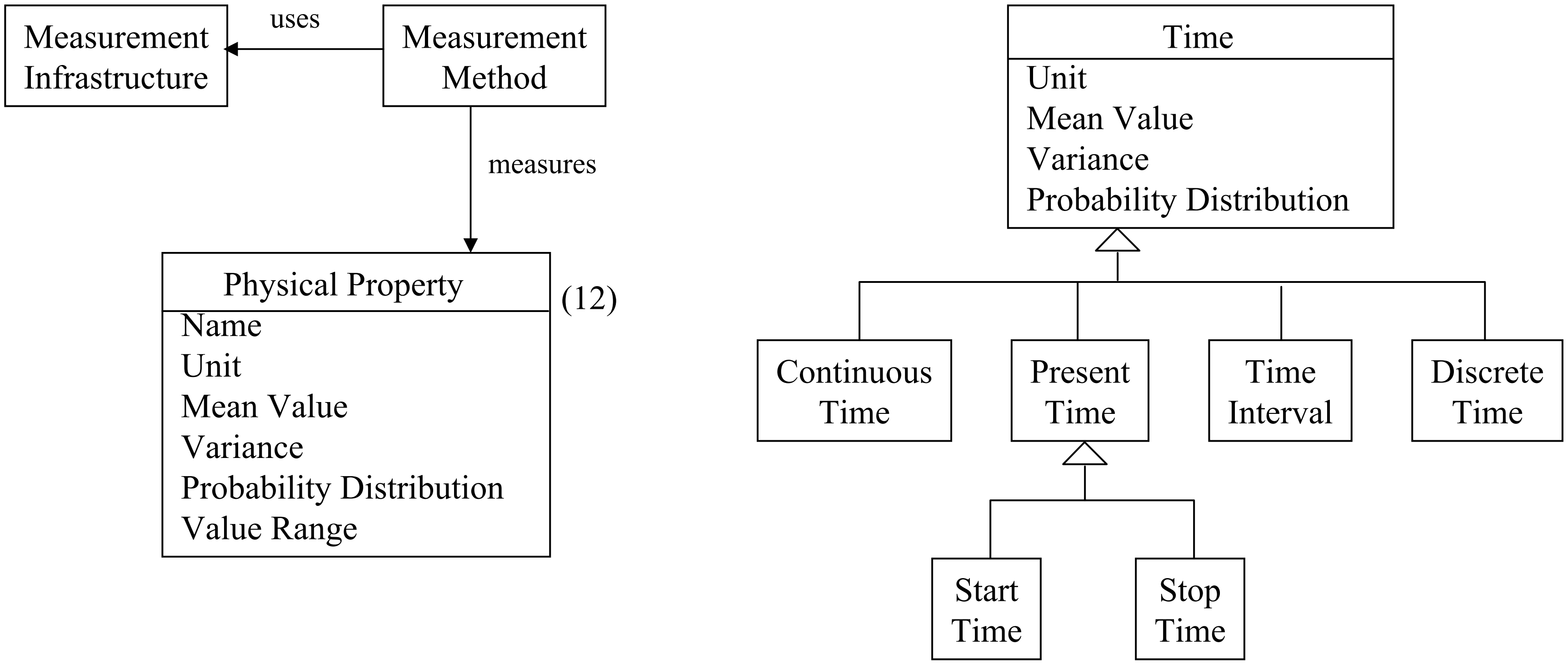
- Stop Time is a Present Time that states where in Present Time a Time Interval ends.

Physical Property and Time

- Discrete Time is time represented by clock pulses of negligible duration. In this approximation events occur on each clock pulse.

Time is one of the most accurately measured quantities that we have. Current accuracy of measurement is about one part in 10 $^{-13}$. Research underway may extend this to 10 $^{-17}$. Many properties now have primary standards based in part on time.

Refined Decomposition for Physical Property and Time, Figure 8.

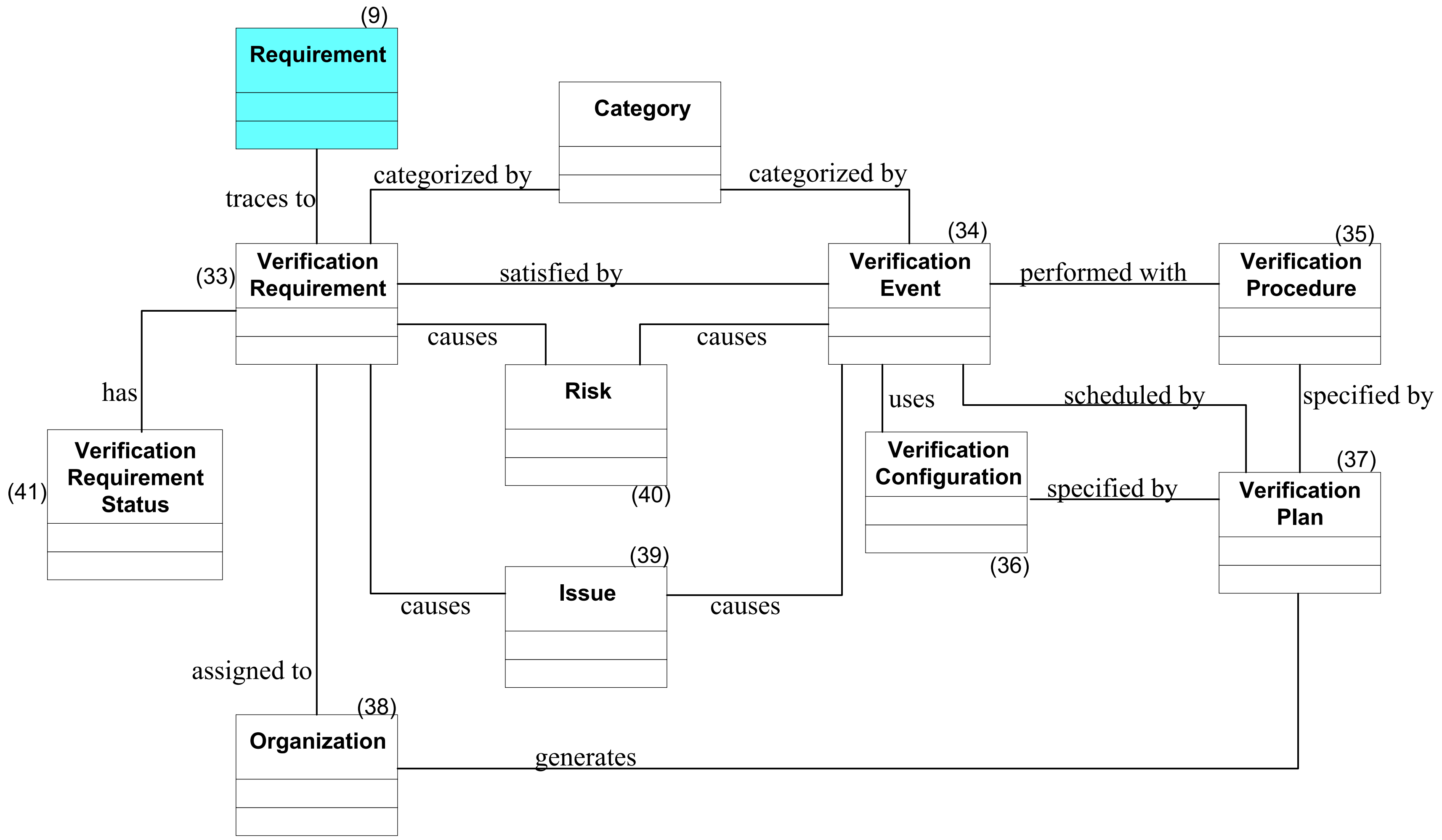


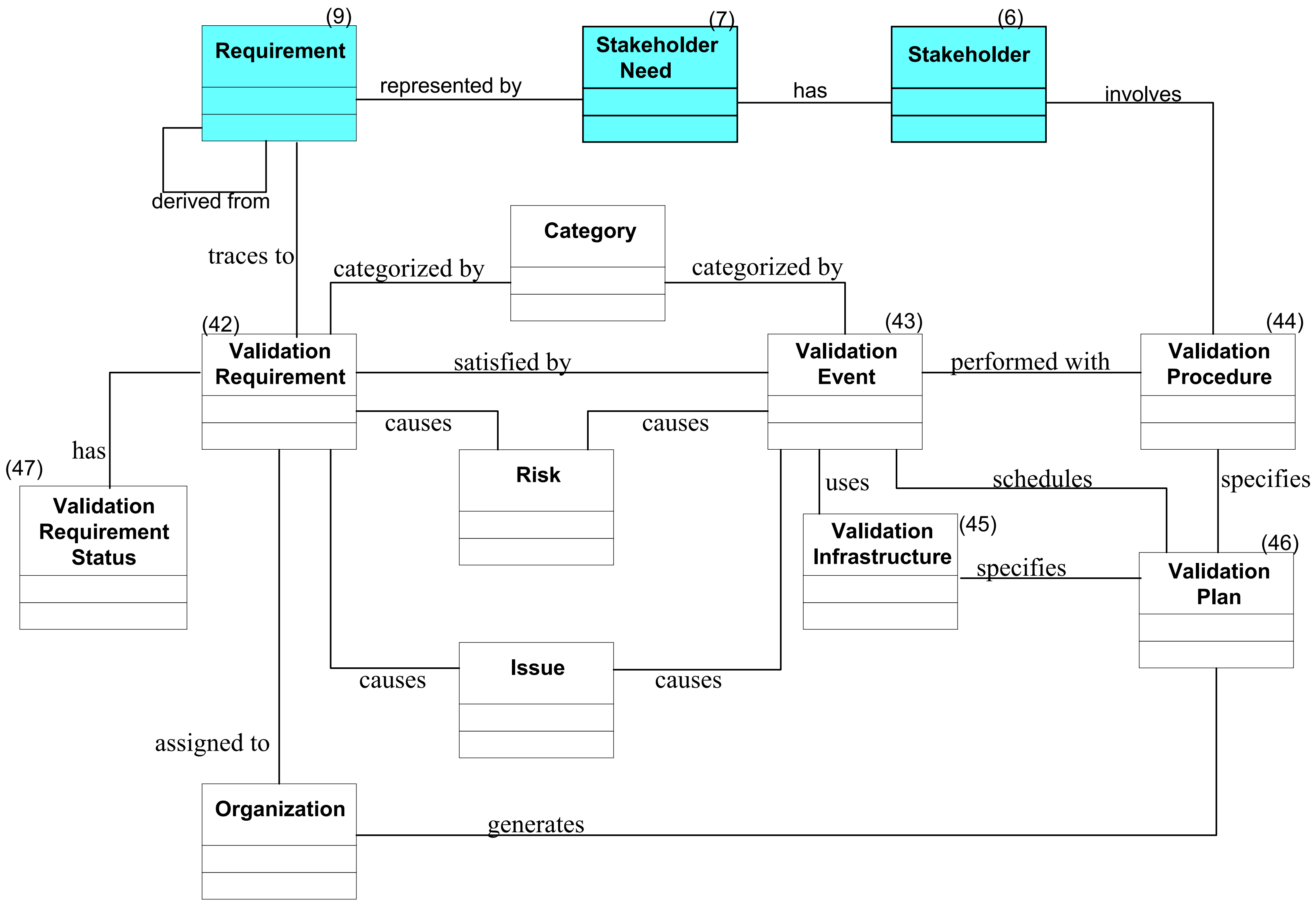
Systems Engineering Management

Three Models follow that are important to systems Engineering Management.

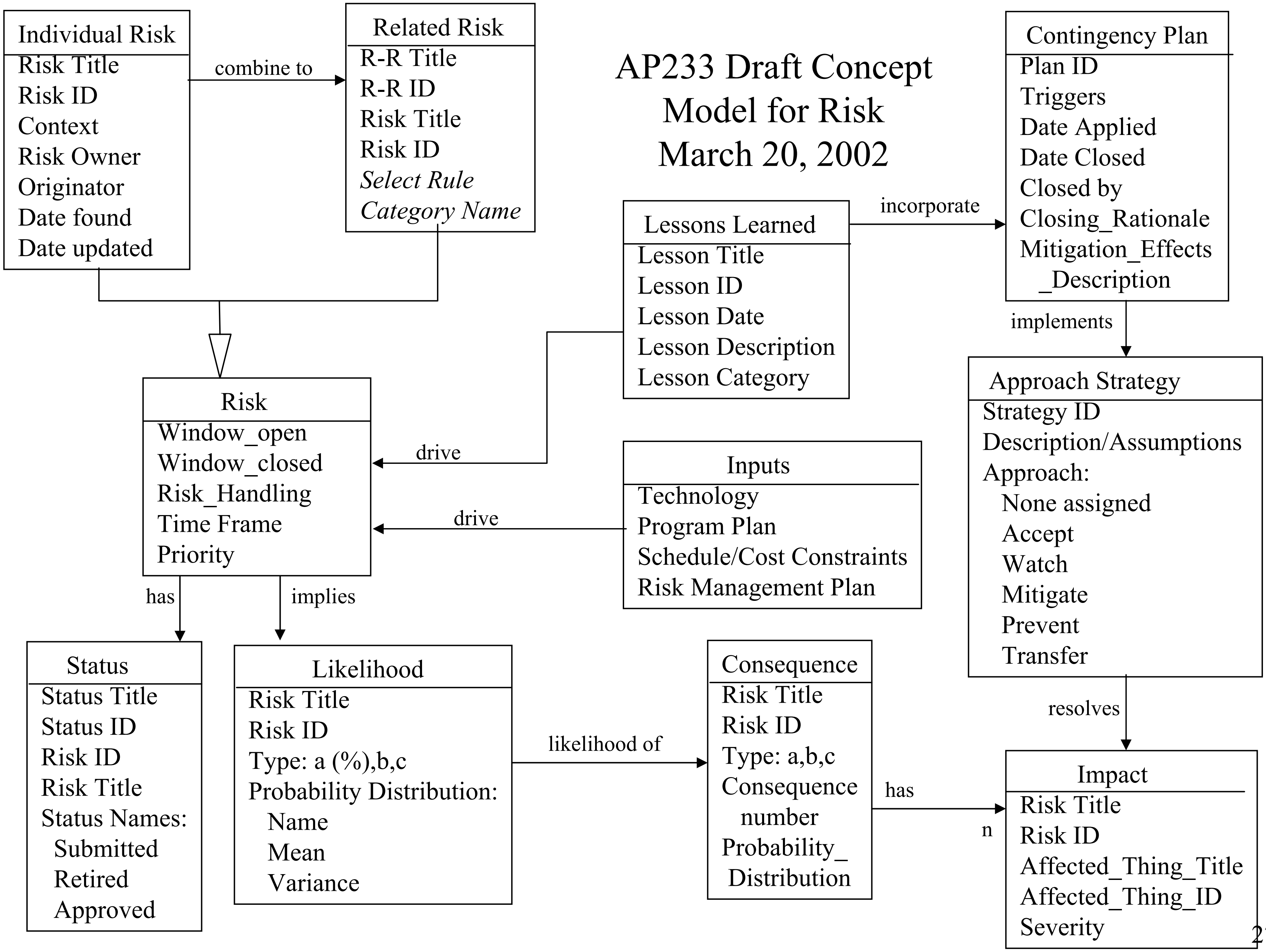
The models for verification and validation are at first draft level and need discussion.

The model for Risk was discussed with the Risk Working Group at the INCOSE 2002 symposium. AP233 is waiting for there corrections and changes. The existing model is based on information from the risk working group, NASA Goddard Risk Attributes in SLATE 'GPM' Data Base March 7,2002 (Dave Everett), and from NASA JPL Risk Process Diagram


(9)
Requirement
Category
traces to
categorized by
categorized by
(33)
Verification Requirement
satisfied by
(34)
Verification Event
performed with
(35)
Verification Procedure
causes
causes
Risk
has
uses
scheduled by
specified by
Verification Requirement Status
(41)
(40)
Verification Configuration
specified by
(37)
Verification Plan
(39)
(36)
Issue
causes
causes
assigned to
(38)
Organization
generates


(9)
Requirement
represented by
(7)
Stakeholder Need
has
(6)
Stakeholder
involves
derived from
traces to
Category
categorized by
categorized by
(42)
Validation Requirement
satisfied by
(43)
Validation Event
performed with
(44)
Validation Procedure
causes
causes
Risk
has
(47)
Validation Requirement Status
uses
schedules
specifies
Validation Infrastructure
(45)
specifies
(46)
Validation Plan
Issue
causes
causes
assigned to
Organization
generates

AP233 Draft Concept Model for Risk March 20, 2002



Category

The decomposition tree for System Assembly is more than a simple parts tree. At any node one may introduce a category of parts. For example, an automobile may have several different engines that can be used in the automobile, each providing a different level of economy and performance.

Categories are a grouping of elements into a set based on defined properties that serve as selection criteria for which elements of all those in the universe belong in that set Explanation: It is categorization that enables us to define alternatives and create taxonomies for libraries. This is one of the forms of generalization/specialization. Note that this is NOT INHERITANCE as found in object-oriented software languages. Physical elements, matter and energy, do not inherit their properties. Rather they posses the properties of themselves and can be identified by measurement of those properties. For a discussion of these issues in computer science see the work of Barbara Liskov and her CLU language.

Note: the subcategories may be exclusive or inclusive and
the subcategories may exhaust the super category or not
there are four such possibilities
Category is the basic concept in the physical world to support specialization - generalization.