Tying Requirements to Design Artifacts

Clark Briggs
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Mark Sampson
UGS
5800 Granite Pkwy Building 1
Suite 600
Plano, TX 75024

Abstract. A new generation of Product Data Management and Product Lifecycle Management tools offer a combination of systems engineering and design engineering capabilities with a common database. This provides the opportunity to tie requirements to elements of the design such as CAD models, drawings and analyses. This connection of the currently disjoint systems engineering and design engineering modelling domains provides a new opportunity for a rich set of relationships and the promise of alleviating much of the tedious checking of requirements usually performed by design engineers. This paper explores the use cases for joining systems engineering and design engineering modelling and proposes a set of relations that provide meaning to the new links.

INTRODUCTION

Integrated Product Design Teams (IPDTs) are a common organizational approach to product design and development in the aerospace and defense industry. These multidisciplinary teams are staffed to address the many aspects relevant to development of the product from problem statement, to design, manufacture and operational use. Systems engineers and design engineers are two engineering specialties in the IPDT that work closely together, iteratively developing the requirements, solution architecture, and system design.

The interface between these two is the area we are addressing. The problem statement for the design team is established by the systems engineers. This consists of the requirements, the functions and perhaps the system architecture. The design engineers establish early configurations, evaluate the performance and provide illumination to the requirements. They go on to completely define the product and develop the fabrication specifications.

We are interested in following a two typical, high value and intensive interactions between systems engineering and design: 1) establishing a requirements set to be used by design engineers to guide the product definition and 2) requirement verification during design completion and review.

At the design gates, the state of the design is reviewed prior to moving into the next phase. In these reviews, the requirements are reviewed, the design is reviewed and the design is compared to the requirements.

These practices are well known and in use in many aerospace and defense organizations, although implementation can be quite manual. For example, the design engineers might be handed the lower level requirements extracted by the systems engineers from the requirements tool used by the systems engineers. The design engineers keep, amplify and interpret these, managing them in their engineering notebooks. At the design review, the design engineers prepare tables that show where the requirements they use came from, and how the design meets them. This can entail quite a bit of work prior to the review and this section of the review can

take 1 to 2 hours. Our proposal is that an integrated tool can better carry this process, relieving the engineer of a tedious, manual and risky activity.

A basic tenet of the approach is to link objects and have the link carry meaning. There are a wide variety of objects in the union of systems engineering and mechanical engineering and so a wide variety of meanings can be given to the links. We illustrate potential concepts with use cases and candidate linking relationships. It is these links that give power to the resulting software system.

The eventual software system will be used by IPDTs, in particular the system engineers working with the design engineers. Of course, the difficulty presently is that few tools contain both the systems engineering and design models, and that interfacing existing single domain tools is difficult and costly.

THE GOAL

The product development process usually begins with input requirements such as customer specifications, standards, lessons learned, telephone conversations, etc. We start the process by capturing these. When the customer says we need to do X, we start deriving new requirements—if we need to do X then we also need to do this and this, which leads to new requirements. These in turn lead to other requirements.

This “explosion” of requirements—a single requirement leading to thousands—is why you need systems engineering tools for organizing these requirements. This pool of requirements is where the current requirement management tools live.

As we develop these requirements, we start developing a list of features or functions that need to be built into our product. We also start looking at alternative ways of accomplishing what we need to do. While we are doing this, we need to consider other views of our system. See Figure 1. We need to able to test it, make sure we can distribute our product, that we can support it in the field, that we can train people to use it, that we can manufacture it, that we are using the right materials, that our supply chain can support it, etc. You can see here we are creating relationships between these different views so that the materials people, for example, can see what requirements are driving them because their material is used in a test set, which verifies a particular component, which performs a particular feature, which is driven by customer requirements. Because they are related to each other, we can see how requirements flow throughout the entire product development process. This way, the materials people are participating in the same product development as the rest of the team because the requirements are connected to them.

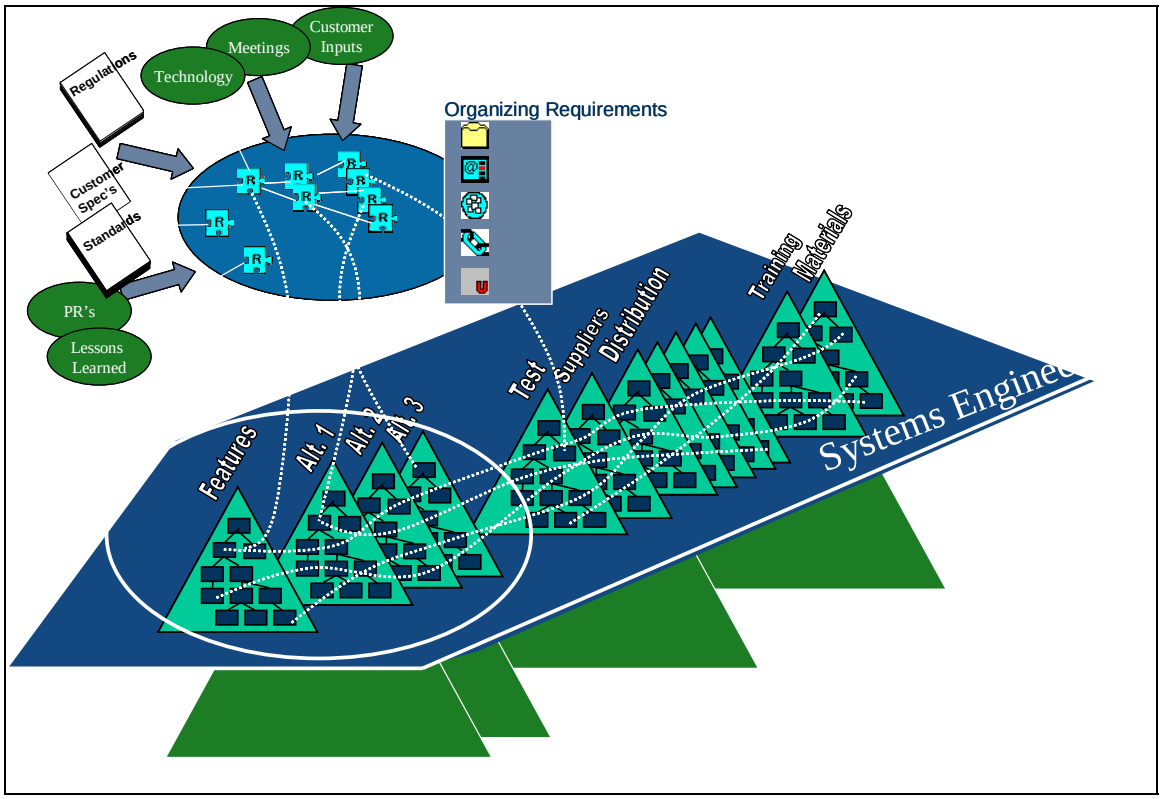

Figure 1. Integrated Systems Engineering and Design Engineering Models

EXAMPLES OF LINKS AS UNDERSTOOD CURRENTLY

A commonly cited example of linking requirements to design is to make a link to drawings in the Mechanical Computer Aided Design (MCAD) Product Data Management (PDM) system. This might convey any of several meanings and the appropriate meaning might change as the design matures.

The drawing goes through a life cycle, at some point becoming released and possibly being revised. As an indication of the conundrum we address in this paper, consider the question "Should the meaning of the link change when the drawing is released?"

To appreciate the difficulty with the simple concept of linking requirements to drawings, consider the following example. In subsequent sections, we will develop specific implementation proposals that address these issues.

Consider a geometric design requirement such as

> "The Elbow Joint of the Instrument Arm shall have an operational angle of +/-75 degrees from straight."

This requirement will exist somewhere in the requirement hierarchy. Since it is geographically narrow and specific to a single assembly's function, it will be well down the tree and potentially a leaf. The current notion is that a simple un-named, un-differentiated link or tie to a drawing is to be created.

Let's look at how the link might be used and examine the candidate meanings the link might carry in these use cases.

Link Creation

At some point in the design process, either the systems engineer or the design engineer will

be ready to create the link. This low level design requirement might not be known at proposal or conceptual design time, although the body of the requirement might be available with a "TBD" for the value of the angle. Most likely, the requirement and an angle value will be available at the end of preliminary design.

The drawing, however, is created quite late in comparison. The project epoch "design release" or "release to manufacture" signifies completion of the drawings. For this example, assume the drawings are in place during the latter stages of detail design.

To which drawing should the link point the requirement? This angle is best displayed on the Instrument Arm Configuration Drawing so that both the Upper and Lower Arm are displayed and the angle, along with its extreme values, can be called out. With such a drawing, an engineer could review the configuration, find the angle and determine the limit values by inspection.

In creating a link between this design requirement and this configuration drawing, what does the engineer mean to imply? A minimal meaning might be "the implementation of this requirement can be found in this drawing by inspection." A more concise link label might be "Is Implemented In" but it more correctly means the requirement is implemented in the assembly displayed in this drawing.

A link meaning of "Is Satisfied By" is more troublesome. The engineer, in checking requirements, could begin at the requirement, follow the link to the configuration drawing, but would have to inspect the drawing, find the specific view showing the angle and its limits, and compare the limit values to the requirement values.

"Satisfaction" is true only if the engineer's study of the drawing leads to that conclusion. In fact, the design may not meet the requirement at the early stages of design evolution. This could be quite misleading at worse, or at best, require engineers to remember the link name isn't its value. Thus, over the life of the evolving design, engineers would need to revisit the drawing to determine by inspection whether the design "satisfies" the requirement.

Requirement Verification

At some point, perhaps in late detail design in preparation for the Critical Design Review, several forms of checking will happen. The fact that all the requirements have been captured in the design will be checked and the design will be checked against the requirements.

With an implementation that links the requirement to the drawing, much of the current manual effort and risk of administrative errors is reduced. Considerable thoughtful effort remains, though, along with the potential for misunderstanding.

In preparation for the review, the engineer might go one-by-one through the requirements, follow the link, open the drawing and determine a result by inspection. The presentation material content and review time is essentially unchanged, but the effort and risk of administrative errors is reduced. In this use case, the meaning of the link is more like "Can Be Found In."

The presence of the link (and correct creation of the link) supports automation of the check that the requirements have been captured in the design. In this case the meaning is more like "Is Assigned To." Little more information value than that can be implied however.

Thus, the tying of requirements to drawings with an un-named link can reduce manual effort and administrative errors, but does not add substantial information value. In fact, it can increase misunderstanding because of the multiple unstated potential meanings of the link.

The real value comes from creating specific link meanings that tie to low level design aspects. Further, as we will see in subsequent sections, the link needs to carry computational capabilities to track satisfaction. With this in place, not only is manual effort and administrative

error reduced, but time spent reviewing and checking can be reduced. During the design review, the engineer can simply display the report summary of satisfied and not satisfied requirements. The review board does not need to check the engineer's methodology, nor check samples of the results to determine correctness.

MEANINGFUL RELATIONSHIPS

This section puts forth a variety of names for links. We begin with listing terms commonly used in conversations. This is a rich vocabulary but lacks any significant definition or clarity. Then we illustrate common administrative relationships that must be present in tools to support the broad set of infrastructure required but which aren't relevant to the current discussion. We close with a survey of relations from the few existing example schema which have some system engineering and design content. The following section shows how the richer information model we seek can be used.

Is Allocated To	Supports/ Is Supported By	Is Basis For/ Is Based Upon	Is Implemented By/ Implements	Relies On
Uses/ Is Used By	Is Satisfied By/ Satisfies	Contributes To	Is Necessary For	Depends On
Verifies	Is Derived From/ Derives From	Is a Copy Of	Contains	Is Rendered By/ Renders
Manifestation	Is Represented By/ Represents	Is Linked To	Is Assigned To	References/Is a Reference Of
Has Attachments	Is Member Of			

Table 1. A Sample of Relevant Relations

Table 1 is list of commonly used relations. In most cases, they are bi-directional with slightly different syntax in each direction. Most relationship names act like verbs and an object acts as the noun/subject and another object acts as the noun/objective. In this paper, we will use notions such as left hand side for the subject and displays such as:

left hand side: - relation -> right hand side

which suggests that the relation is the name displayed on an arrow from the subject to the object.

Existing design tool schemas have a wide variety of linkage mechanisms and many provide, from our current point of view, administrative semantics. For example, a file server has folders and files. Although it is not explicit, the relation between a folder and its files is "contains." In the Teamcenter Enterprise Product Data Management (PDM) system, the relationship between a Container and a DataItem is explicitly "Contains." Teamcenter Enterprise also has a "Is Copied From" relationship that is used for tying Copies to their Source. (UGS Corporation, 2005) Other administrative connections are simply named after the objects connected or are used to show a condition. Teamcenter Enterprise has "Item to Vis" to connect a visualization file to its hardware Item and "Has Visualization Files" between more generic documents and 2-D drawings and images. The Vitech CORE implementation of the Department of Defense Architecture Framework (DoDAF) schema says a Defined Term "Is Used In" a Document. (Vitech Corporation, 2004 and 2005)

A few schema exist that contain some systems engineering and design artifacts. Table 2 shows part of the Vitech implementation of the DoDAF from CORE. (Vitech Corporation, 2004 and 2005) The DoDAF schema object set is of modest size and includes Requirements, Functions and Components. The requirement hierarchy relates lower level Requirements to

higher level Requirements through “Is Refined By.” The physical hierarchy of Components is tied together with “Is Built From” going down and “Is Built In” going up.

	Requirement	**Function**	**Component**
Requirement	refined by/ refines	basis of	specifies
Function	based on	decomposed by/ decomposes	allocated to
Component	specified by	performs	built from/ built in

Table 2. CORE DoD Architecture Framework Relations

The connection to the physical architecture says a Requirement “Specifies” a Component and a Component “Is Specified By” a Requirement. The Teamcenter Engineering data management system doesn’t have systems engineering objects but has a “Specification” relationship which is used to associate data such as Standards to which the Item must adhere. (UGS Corporation, 2004)

	Requirement	**Function**	**Component**
Requirement	«derive»		
Function	«satisfy»		
Component	«satisfy»		«assembly»

Table 3. SysML Relations

SysML and UML2, which SysML extends, have a very general object and relation schema more attuned to the software development domain where UML originated. (SysML Partners, 2005) See Table 3. SysML adds the Assembly object which is a container for Parts. The Requirement hierarchy uses “Is Derived From” which means a child refines or restates a parent Requirement. “Satisfies” is used to indicate a dependency relation between a Requirement and a Part. UML and SysML don’t provide standard names for relationships and expect users to provide problem-specific names for the actual relationships used in the model.

AP-233, the STEP schema for systems engineering, provides a generic structure for a wide variety of systems engineering objects, but depends on other STEP APs, such AP-214, to define the physical structure. (Bailey, 2001) For example in AP-233, entities in an Assembly are known as system_view_definitions which are linked together with system_assembly_relationships. Like SysML, AP-233 expects users to provide problem-specific names for the actual relationships.

NEW WORKED EXAMPLES

The following continues the discussion of tying requirements to design artifacts, but considers various types of requirements. The goal is to develop link meanings and to further specialize the implementation notion of design artifacts.

Functional Requirements

Functional requirements might well be “Allocated To” a Function which is “Implemented By” some hardware. For example, “The message processing subsystem shall decode and verify all incoming messages” might be allocated to a Function Block named Decoder which is one

Function Block in a Message Processing Subsystem Functional Block Diagram. See Fig. 2. This Decoder Function Block might point to a Bill of Materials (BOM) Item which is the top level assembly in the mechanical CAD model for the chassis that holds the Decoder. That relationship is "Is Implemented By." See Fig. 4. If the model doesn't use the Function Block, the Requirement is related to the BOM Item with the "Is Implemented By" relationship.

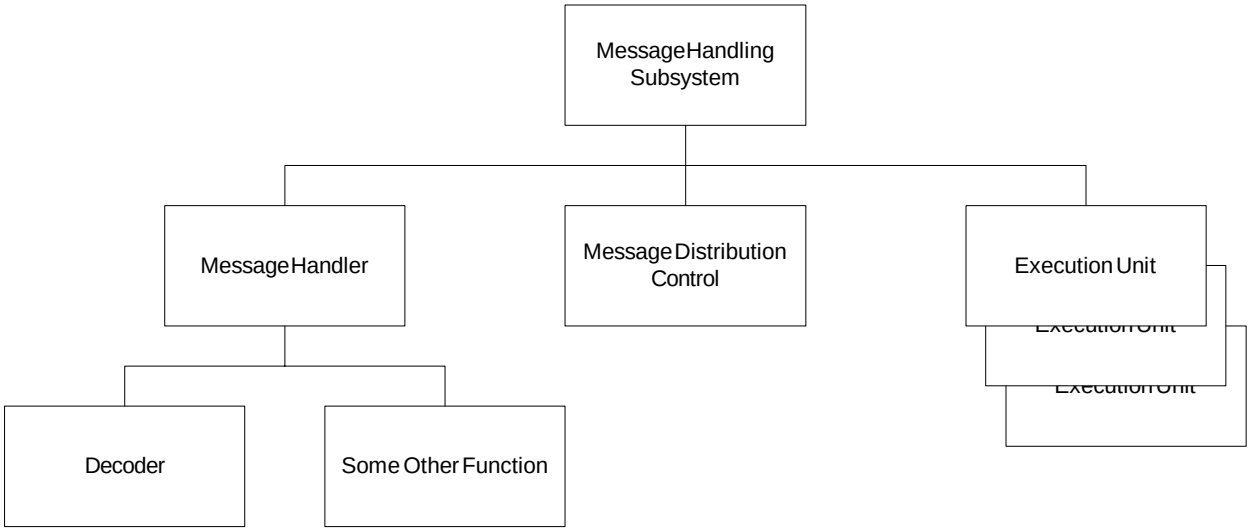

Figure 2. Message Processing System Partial Functional Block Diagram

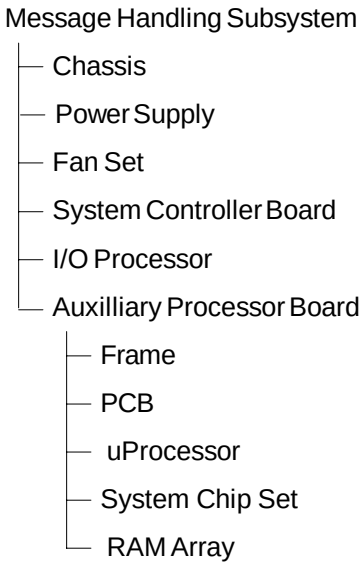

Figure 3. Message Handling Subsystem Bill of Material

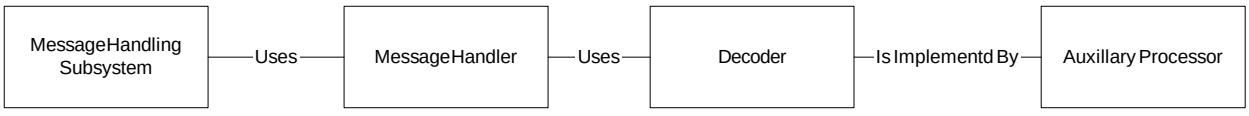

Figure 4. Drilling Down to the Decoder Implementation

What can you learn by following this small graph up and down? Suppose the Decoder BOM Item is part of a higher level assembly that is the Message Handler. That is, the Message Processing Subsystem has a Message Handler, a Message Distribution Control and multiple Execution Units. The Decoder is a part of the Message Handler. Each of these is one or more boards in the hardware set that implements the subsystem. See Fig. 3.

Suppose the question concerns the design capabilities required on the board that hosts the Decoder. Start with the Message Processing Subsystem, drill down its assembly hierarchy in the MCAD models and find the board named Decoder. Find the relationships named "Is Implemented By" for which it is on the right. Follow those to the left and find a Function Block or one or more requirements. See Fig. 5.

Figure 5. Finding a Processor's Function

Suppose the question is where is the requirement reflected in the design? Start with the requirement and find the relationships for which it is on the left. There might be many such relationships, but look for the two used here. If you find "Is Allocated To," follow it to find the Decoder Function Block. Inspect the Decoder to find its relationships and look for the "Is Implemented By." The right side of that should be a BOM Item which answers the question. It could have been that the Requirement itself had an "Is Implemented By" relationship, in which case, you can get to the Decoder BOM Item directly.

Suppose you are studying the Functional Block Diagram for the message Processing System. You can find the requirements that have been "Allocated To" its Function Blocks. In the Message Processing Subsystem Functional Block Diagram, start at the top Function Block and drill down to the Decoder Function Block. Check the relationships for which it is the right side, looking for "Is Allocated To" relationships. One of these should have a left side that is the requirement for a decode function. If you wanted to know which BOM hardware Item implements this requirement, look for "Is Implemented By" relations on the Decoder Function Block and find one that points to the Decoder BOM Item.

Performance Requirements

Suppose the requirement on decoding throughput reads "The decoder shall be able to sustain 100 messages per second." This looks like a performance requirement. We need to be able to drill down to an analysis report on the Decoder BOM Item and find a field with the computed throughput. A good name for that pointer/relationship might be "Is Supported By."

Notice that performance requirements like this might be demonstrated in the Functional model. That is, the Functional Block Diagram might have a behavioural simulation and the analysis report from the behavioural simulation might produce a value for comparison to the performance requirement. This value isn't different in any substantive way than the analysis based upon the hardware design. The latter may have more fidelity, but neither is a measured value, both are predicted values.

Performance Requirement: - Is Supported By -> Behavioural Analysis Report Result

Performance Requirement: - Is Supported By -> Hardware Analysis Report Result

Performance requirements might point to analysis reports that predict the performance, but there might be tests that measure the performance. The test article might be the production item or it might be some test article, prototype, or engineering model. It may be a close representation of the production article or it might even be a production article (in the case of a proof test). The test articles might only be partially representative of the design since they might be testing only part of the system or performance. The test setup might only test part of the operational behaviour, such as a vibration test for launch loads. One point of this is that test data is no more "proof of performance" than analysis. The relationship between a performance requirement and an analysis result should be the same "Is Supported By" meaning as between a performance requirement and a test result.

Design Requirements and "Is Implemented By"

Consider the geometric design requirement

> "The Elbow Joint of the Instrument Arm shall have an operational angle of +/-75 degrees from straight."

This might well be a derived requirement that was generated from early studies of operational scenarios for the Instrument Arm that considered stowing, reach and coverage for sampling and delivery of the sample to the on-board processing unit. So it will be part of the requirement hierarchy somewhere below a top level requirement such as

> "The System shall support a Sampling Instrument Package."

This is the type of requirement that we would really like to automate the management of. It's a good one for the "Is Implemented By" relation:

> Requirement: - Is Implemented By -> BOM Item

For this example, the BOM Item would probably be the Instrument Arm top level CAD assembly. It could be some higher assembly such as the Instrument Package or even the System, but it becomes less helpful. We would actually like the right hand side of this relation to be a feature in the Arm BOM Item MCAD model.

MCAD assemblies can have a variety of structures and the angle of a joint that supports the Upper Arm and Lower Arm could be in a couple of places. It could be in the mate constraint between the Upper Arm and the Lower Arm or it could be in a skeleton or active layout.

At any given time, this angle could have a value between its limits that is set by the configuration. We really want to point the requirement at the limits, not the current value. Further, this requirement must be broken into two requirements:

> "The Elbow Joint shall have an inner operational angle limit of +75 degrees from straight."

And

> "The Elbow Joint shall have an outer operational angle limit of -75 degrees from straight."

Both of these can be found in the upper and lower limits of the constraint, either in the skeleton or the mate. The constraint in the skeleton might be a different feature type than that of the mate.

It is a human that makes these connections. Suppose the designer has constructed the MCAD model which is the BOM for the Instrument Arm. A system engineer, a mechanical system engineer or the designer might be connecting the requirements to the BOM. Finding the appropriate right hand side for the relation is a visual activity. Given the pair of limit

requirements, the engineer drills down through the Instrument Arm MCAD model to find the feature or mate. Notice that several things must be modelled in a consistent manner. The angle coordinate system in the MCAD model must be zero when the Lower Arm and the Upper Arm are straight. The angle must be positive when the arms fold in and negative when the arms fold out. Here, in and out are conventions that the design team must adopt. The engineer making the connection must recognize and check these. If the expression of the requirement and the design of the MCAD model don't match, things must be fixed.

A COMPUTATIONAL CAPABILITY FOR LINKS

In addition to describing the meaning of the link, the link might carry state or value. The most common place to think of this is the "Is Satisfied By" link between a requirement on the left and a physical implementation on the right. As mentioned in a previous example, simply naming a link "Is Satisfied By" is misleading since there is no guarantee that the design element carried by the right hand side in fact "Satisfies" the requirement. At various stages of the design life cycle, the design may not, in fact, meet this requirement. Unless meanings like this are augmented by a value, the meaning is no more than "satisfaction can be determined by manual inspection of the contents."

In this section, we explore the nature the computational capability required and the consequences of providing it.

If links are to carry a value in support of an "Is Satisfied By" relation, the simplest computational point of view is the link acts like a relational operator which returns a True or False result. "Is Satisfied By" has this behavior. The angle requirement on the Instrument Arm Joint read:

> "The Elbow Joint shall have an inner operational angle limit of +75 degrees from straight."

The link was created perhaps by the systems engineer or the design engineer by interactively drilling down to the Instrument Arm Configuration Drawing and finding the Angle Upper Limit call out on the drawing. The MCAD system clearly knows the numeric value of this Angle Limit even if it is displayed as text on the Configuration Drawing.

The requirement is typically stored as a text string since current practice is that all content evaluation is provided by human interpretation upon reading. If, in the link creation activity, the engineer selects the text of the value (ie, "+75") as the left hand side, a computational comparison is possible. The link computational processor can use normal arithmetic operations to promote the string to a numeric value for comparison to the right hand side numeric value pointed to in the Configuration Drawing.

But what is the computational relation? In this case, the implementing engineer understands the requirement and how it is mapped to the configuration and can choose from the usual pallet of relational operators such as "<", "<=", "=", "=>", and ">". A strict interpretation of the requirement calls for an equality check as in "=" but this is computationally and logically risky. The engineer needs to seek clarification and refine the requirement to something more readily implemented such as:

> "The Elbow Joint shall have an inner operational angle limit of not less than 75 degrees from straight."

With this clarification, the operator "<=" is appropriate. If the design currently has the Angle Upper Limit set at 77 degrees, the resulting expression reads:

> The Elbow Joint inner operational angle limit of 75 degrees is less than or equal to the Angle Upper Limit call out of 77 degrees.

The computed result will be True which means the design "Satisfies" the requirement.

Should the design change to where the Angle Upper Limit falls to 73 degrees, perhaps due to a volume conflict in the elbow or a collision at the Lower Arm outer end when stowed in this inner limit position, the computed result will be False. This signifies that the design does not "Satisfy" the requirement.

Thus, the system could always provide the computational result with the link name. This both avoids misunderstanding and significantly reduces the effort required to determine if the design "Satisfies" the requirement. In particular, it is not necessary to open the model in the MCAD tool or know how to use the MCAD tool. Further, report operations in the PDM can tally the status of "Satisfaction" of a set of requirements without further mechanical engineering domain knowledge.

As noted, during the early stages of development, the requirement might have a "TBD" value as in:

> "The Elbow Joint shall have an inner operational angle limit of not less than TBD degrees from straight."

If this value is present in the requirement when the engineer is creating the link, the engineer should proceed as before selecting the text "TBD" for the left hand side. The relation processor will not be able to convert this to a numeric value and should produce a result of the comparison like "Unknown." Such tri-state arithmetic, as opposed to two-state or Boolean arithmetic, is readily accommodated. This enables PDM reports that tally the number of "TBD" requirements which is a common metric for design completion during the preliminary design phase.

Performance requirements can be implemented in a similar fashion. A requirement might be:

> "The decoder shall be able to sustain 100 messages per second."

The subsystem analysis report will contain a variety of information supporting and substantiating the analysis. Somewhere in the report, text such as the following will exist:

> "Using the model described above, the worse case throughput results for the 3 required scenarios are as follows.

Case	**Messages per Second**
Case 1	121 mps
Case 2	107 mps
Case 3	144 mps
Limiting Speed	107 mps

Table 23. Case Summary

> As shown in the table, the decoder worse case speed over these scenarios is 107 messages per second."

When making the link between the requirement and the analysis report, the engineer will choose the requirement text "100" for the left hand side and the text "107" from the row of the results table labeled "Limiting Speed."

Other discussion in the text might lead to a lower value, for example, when factors of safety are applied and that text can be chosen instead. Looking ahead to updating the analysis, it is desirable to pick up text fields that the report generator will recompute to avoid manual steps. For example, suppose the report went on to say:

> "Applying a factor of safety of 1.10, the current design does not have enough speed margin."

With this text, the desired value isn't even present, and, if it was, the report generator isn't likely to recompute it.

This leads to a desire that the link computational engine allow expressions in the left and right sides of the relation. Thus the requirement field might be compared to an expression made from the table entry and the factor of safety. It would be helpful if the text value of the factor of safety in the preceding paragraph could be labeled with the name "Factor of Safety" and the table entry labeled with "Worse Case Value" such that the link relation right hand side could be the expression "Worse Case Value / Factor of Safety." Thus the report generator would update the table and the link computational engine could recompute the value of the link condition.

BUILDING REQUIREMENTS INTO THE DESIGN

Modern MCAD systems are parametric and associative. This allows related geometric features to depend upon each other such that aspects of the design are automatically enforced. Mating conditions such as point-in-plane and axes-collinear implement geometric design intentions. Dimensional constraints such as making the length of a horizontal side equal to half the length of the vertical side also express design intentions.

Consider again the Instrument Arm Angle Inner Limit requirement:

> "The Elbow Joint shall have an inner operational angle limit of not less than 75 degrees from straight."

When the design engineer is constructing the skeleton layout that drives the Configuration Drawing, a construction line is usually built to represent the limit and its angle to straight is manually entered into the construction. In a parametric MCAD system, the engineer can access that value, change it to "70" and the line representing the limit will be repositioned. In the prior discussion on computing the "Satisfaction" of a requirement, this value was the right hand side of the relation.

Suppose instead of manually entering the value "75," the engineer tied the value to the requirement. This might be done interactively, when the construction line's angle value field is open for editing, by drilling down to the requirement and selecting the text field "75" in the requirement.

The result, then, is that the geometry of the design is dependent upon the requirement.

This changes the entire notion of linking requirements to the design artifacts. Artifacts are built out of, or built upon, requirements. Such requirements can never be violated. There is no need to check for "Satisfaction." Furthermore, should the requirement be changed, say to an inner limit angle of "72 degrees," the geometry engine will update the design.

This leads to multiple approaches to using requirements in the design of the Instrument Arm configuration and components.

- The limit value can be a static call out on the configuration drawing and the requirement text can be checked against it.
- The limit value can be tied to the requirement text and driven by the requirement text. Checking the angular position of the limit is never required.
- The geometry engine can use the Assembly Configuration to determine the inner limit by, for example, driving the angle until the Elbow binds or the outer end of the Lower Arm hits something. This computed value could be checked against the requirement text value.

DESIGN CHANGES AND MODEL UPDATING

Building associative geometry models is a significant labor saving capability of current MCAD systems. The design engineer builds design intent into the constructed geometry in such a way that the design is durable to certain changes. This capability has been implemented into

some analysis tools which can automatically accommodate changes to the analyzed geometric design.

Suppose a component stress requirement was:

> "Under loads of 20g in any direction independently, the stress shall be less than the material yield stress with a factor of safety of 1.20."

The analyst would build a finite element model from the design geometry, make multiple load cases covering the stated "any direction" requirement, and capture the worse case stresses in a table in the report. When the part design geometry changes, the analyst can use the analysis tool to accept the design model changes, rebuild the mesh, resolve the equations and update the report. Depending upon the severity of the design geometry changes, the analyst may not need to do anything other than accept the automatic changes and authorize the update run.

Using the prior requirement computational capability, the requirement engine could re-evaluate the "Satisfaction" of the requirement. The requirements maintenance engineer might trigger the engine to re-evaluate the requirements when an update notice was delivered indicating that the design artifact on the right hand side of the requirement had been changed. Given that the fields in the analysis report had been labeled and the labeling preserved when the Analysis Report Wizard reconstructed the analysis report, the requirement engine could readily recompute the "Satisfaction" metrics.

Of course, the changes could have been so significant that associativity was damaged. The analyst might have changed the analysis report structure and deleted the body sections with the labeled result fields and not tied the requirement right hand side back to some other appropriate text to repair the damage.

In another change scenario, the requirement might have changed. The analyst would be notified that a driving requirement had changed and, upon study of the change, altered and rerun the analysis. As above, it could be that the requirement changed in simple ways that did not require manual intervention or repair. But it could have changed in structure such that the analysis was significantly altered. The analyst would probably need to rebuild the link between the requirement and its value fields and the analysis report and its value fields.

In either case of a changed requirement or a changed report because of a design change, the PDM reporting system can tally the number of changed requirements that had not been reflected in design changes or analysis changes. It could report the number of changed requirements that resulted in new "Dissatisfaction" values after the analysis reports had been updated.

LIFE CYCLE CONSIDERATIONS

MCAD and PDM systems provide configuration control for the design CAD models as well as the analysis models. This is done by making the BOM Item revisionable. That is, the BOM Item has alpha numeric descriptors to denote its version. Version numbers increase as the design evolves. At appropriate change events, the design engineer will close the current revision which captures the state of the design before engaging in the design change. The next revision will be used to capture the evolution of the design as the design team attempts to accommodate the required change.

The design engineer can use the PDM to control the state of a design session. The design BOM can be reconstituted as of any prior revision, or design on certain portions of an assembly can proceed while other portions are held at a fixed revision. The latter effectively isolates one part of the design team from changes induced by another part of the team. The lead design engineer will then call for reconciliation of such changes at epochs known as "baselines."

It will undoubtedly be the case that the requirements hierarchy will change at a different pace

than the design. The requirements engineers will be actively refining the requirements and, at occasions that suit their progress and milestones, establish revision baselines.

The design engineers will similarly be evolving the design, establishing revisions and baselines as suits their environment. To isolate the two groups from daily changes, the design engineers would work to a given requirements baseline. Thus, for short periods, the requirements to which they work would not change and the design team could focus on achieving a given design that more or less appropriately meets the requirement baseline.

To do this, the design engineers would use revision rules that configure the requirements database to the current requirements baseline. The link computational engine would use the requirements database configured to that requirements baseline to determine the left hand side of the link relations. Similarly, the design BOM would be configured by revision rules that picked the current design baseline. The PDM reports that captured metrics would be associated with this requirement baseline and that design baseline.

Suppose the design team completed a design epoch and created a new design baseline. To continue working to the evolved, but not considered, requirements, the requirement revision rules would be changed to use the requirements baseline built from the next requirements revision. The link computational engine would recompute the "Satisfaction" metrics and the PDM report engine would show the design team where the design "Satisfied" or failed to "Satisfy" the evolved requirements.

CONCLUSION

By implementing systems engineering and design engineering tools in a common database, significant improvements can be found in tying requirements to design artifacts. Connecting them with simple links provides some labor savings and risk reduction, but more substantial savings will result from implementing a semantic net using named relationships. This allows users to automatically detect and verify compliance. We illustrated this with the "Is Satisfied By" relationship for several design and performance requirements.

Providing computational capabilities for links can lead to significantly reduced manual effort and misunderstanding. Extending the notion of associativity to these links will let the PDM report engine provide metrics that track the up-to-date status of the database. Going further to build the requirements into the design basis removes all checking and drives the design from the requirements.

REFERENCES

Bailey, Ian, "STEP for UML Users," Eurostep Group AB, 2001.

SysML Partners, "Systems Modeling Language (SysML) Specification," Version 0.9, 10 January 2005.

UGS Corporation, "Teamcenter Engineering Object Model, " 2004.

UGS Corporation, "Teamcenter Enterprise MODeL Reference." Publication Number MT00323 H, July 2005.

Vitech Corporation, "CORE® System Definition Guide," Version 5.0, Vienna, Virginia, 2004.

Vitech Corporation, "CORE® Architecture Definition Guide." Veersion 5.1, Vienna, Virginia, 2005.

ACKNOWLEDGEMENTS

This research was carried out in part at the Jet Propulsion Laboratory, California Institute of

Technology, under a contract with the National Aeronautics and Space Administration.

BIOGRAPHY

Clark Briggs is the Hardware Development Process Engineer at the Jet Propulsion Laboratory, California Institute of Technology where he is responsible for the mechanical design process and the mechanical design tool service. He received his doctorate from Columbia University and his B.S. from the U.S.A.F. Academy. Clark is a member of the International Council on Systems Engineering and an Associate Member of the American Institute of Aeronautics and Astronautics. His contact information is available at http://eis.jpl.nasa.gov/~briggs.

Mark Sampson is the evangelist for integrating systems engineering and requirements with the product lifecycle for UGS.

Mark earned his B.S. in Computer Engineering from BYU and his M.S. in Systems Engineering/MIS from USC. His education and 20+ years of work experience at GE, TI, and other organizations exposed him to CAD system development and support where he quickly realized the need for a system-level CAD/CAE environment to cover the critical early development phases of the product lifecycle. Mark began working with developers at Texas Instruments the early 90's developing a number of patented, systems-oriented CAE features in what they called the SLATE (System Level Automation Tool for Engineers) environment, which went public in 1994. Since that time, Mark has been involved with ongoing efforts to integrate/automate systems engineering through computerized tools and developing standards--a number of which have been built into UGS' Teamcenter solutions.