INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

A HIERARCHICAL REQUIREMENTS MODELING SCHEME TO SUPPORT ENGINEERING INNOVATION

Jonathan R.A. Maier[1], Thulasiram Ezhilan[1], Georges M. Fadel[1], Joshua D. Summers[1] and Gregory Mocko[1]
[1] Department of Mechanical Engineering, Clemson University

ABSTRACT
The practice of requirements capture, modeling, and management is more advanced in some fields, such as software engineering, than is currently evidenced in the field of engineering design. This situation is due in part to the increased number of domains that engineering problems span. Software designers, for example, need not concern themselves with physical working principles, material selection, environmental regulations, safety tests, and so forth. To address this situation, we describe a matrix based hierarchical requirements modeling scheme capturing seven domains of interest to engineering design problems. We illustrate how the application of our modeling scheme can foster engineering innovation through examples drawn from automobile sub-systems.

Keywords: Requirements modeling, matrix model

1 INTRODUCTION

1.1 Frame of Reference
In dynamic markets where technology is evolving along with customer tastes, the ability to satisfy design requirements innovatively is a key competitive advantage. To facilitate this kind of innovation, designers need to be able to explore the effects of using alternate technologies to satisfy high level requirements while meeting system tests. Innovative technologies may embody new working principles, provide alternate functionality, and be subject to different low level requirements than existing technology. For example, a hydrogen powered vehicle must transport passengers and cargo just as a gasoline powered vehicle. However there are different requirements on the safety of containing hydrogen versus the safety of containing gasoline. Emissions standards and tests for a gasoline powered vehicle become meaningless for its hydrogen counterpart, however the hydrogen vehicle may have unique safety issues associated with refueling or battery charging.
A comprehensive modeling scheme is needed that enables designers to capture the domains of interest for their design, visualize how the information in each of these domains is related between each domain and within each domain, and study the effect that changing an entity has on the entities in that domain and on the entities in the other domains.
Following the systematic engineering design methods of Pahl and Beitz [1] we recognize the need to at least model requirements, functions, working principles, and physical components. In addition, in our research we also capture individual component parameters (i.e., design variables), tests, and test measures. Altogether this yields seven domains of interest.

1.2 Review of requirements modeling approaches
One tool that is often used for the purpose of fostering innovation with respect to requirements satisfaction in both management practice and engineering design is Quality Function Deployment, specifically the House of Quality (HoQ). The HoQ a matrix based tool that is most often used to map customer attributes to engineering characteristics. Similar HoQ matrices can be used to study the link between engineering characteristics and process characteristics, etc. [2] However, as discussed above, we recognize the need to capture additional domains of interest to show how the initial requirements motivate the final design choices. A detailed comparison of our proposed modeling scheme with HoQ matrices can be found in Mocko et al. [3].

Many other authors have investigated tools to facilitate requirements modeling. The Product Data Specification (PDS) method advocated by Pugh [4] records and tracks requirements, creating an evolutionary document matching the characteristics of the final design as it develops.

Lubars et al. [5] reviewed the field of requirements modeling, performing a field study of ten organizations to discover how they define, interpret, analyze, and use requirements in the development of software systems and products. That research focused on solving organizational problems arising from mismanagement of requirements.

Easterbrook et al. [6] investigated formal methods to model requirements. They developed an approach to provide augmented cost effective techniques for requirements specification. They dealt with the test requirements for fault preventive systems and the expected behavior of the embedded software systems for fault preventive systems in spacecraft.

Lee and Kuo [7] developed an approach for performing requirement trade-off analysis in complex systems. The requirements classification scheme analyzing the heterogeneous requirements identifies the relationship between the requirements. The requirements that conflict or seem to be irrelevant are analyzed for trade-offs. The researchers used parameterized aggregation operators to combine or perform trade-off analysis of the requirements. Their framework provides a formal way to analyze and model the conflicts between the requirements.

The MOOSE method advocated by Gershenson et al. [8] includes a taxonomy classifying corporate requirements as manufacturing, marketing, service, or financial. These researchers considered the taxonomy to be an organized method of gathering, managing and retrieving the requirements.

Rolland and Prakash [9] argue that conceptual modeling incorporates a broader view of the system information than traditional requirements engineering. Following this perspective, they derive purposeful system requirements that better meet the requirements of the users.

Ramesh and Jarke [10] discuss reference models for requirements traceability; the purpose of which is to reduce significantly the task of creating application-specific representations of systems. According to their method, the user selects relevant parts from a reference model, adapts them to the problem at hand, and configures a specific solution from the adapted parts. Their method uses proprietary software including sub-modules for requirements management, requirements rationale, design allocation, and compliance verification.

Laguna et al. [11] propose a method to specify the requirements for reuse thereby reducing the errors that occur due to incorrect requirements specification. A multi-viewed requirement technique that would be helpful for requirement analysis was earlier developed by Delugach [12]. They develop a metamodel to enhance the requirements re-use approach. Their focus is on correct requirements determination for software development and lifecycle.

Fu et al. [13] studied requirements in relation to the product life cycle. They categorize requirements as Voice of the Customer (VOC), market requirements, statutory requirements, corporate requirements, and realization requirements. They provided a systematic framework for modeling, managing, accessing, retrieving, updating, changing, sharing and reusing the design requirements. They also discuss the impact of design requirements in the product development life cycle.

Clarkson et al. [14] review the existing literature on change propagation and conclude that no existing methods (e.g., from software engineering) are appropriate for complex mechanical engineering systems. They captured the past experience of the actual design engineers as a part of their approach. They introduced a matrix-based method for predicting the likelihood, impact, and risk of changes to existing designs (i.e., variant design). Their approach does not include the effect of change propagation on domains such as the system functions, components, or tests.

Somé [15] introduced a method to narrow the gap between the customers and the system development process. He employed use cases to describe the behavior of a system, and to capture and document the requirements. These use cases supported requirements verification, validation and clarification. They suggested that use cases that describe the possible interactions may be used as an effective way for functional elicitation and analysis.

Traditional requirements engineering addresses different phases of interaction with requirements during the design process. These phases include requirements elicitation, analysis, allocation, traceability, verification, creation of requirements taxonomy, requirements modeling and requirements propagation [16]. Elicitation of requirements refers to the phase of requirements determination in which an initial set of requirements for a system is discovered. It may also be defined as the process of gathering the requirements that govern the product or that are required/has to be satisfied to define the

product. Analysis of requirements is an iterative process involving evaluating, decomposing, sorting, structuring and prioritizing, change processing and approval processing. Allocation assigns the requirements to detailed processes, the detailed elements/components of the system/product. Tracking performs continuous analysis and keeps track of the changes for the process as a whole. Verification is used to check whether requirements have been met. Taxonomy is the hierarchical classification of the large amount of information. Propagation refers to the cascading of the requirements from the system level to the subsystem or component levels. Not all approaches address all the phases in requirements engineering. The table below summarizes the approaches discussed in this section with respect to the phases used in requirements engineering. It is evident that no one approach covers all phases.

Table 1. Comparison of various requirements modeling approaches

Author	Capabilities						
	Elicitation	Analysis	Allocation	Traceability / Tracking	Verification / Validation	Taxonomy	Propagation
Fu	Yes	Yes	Yes	Yes	Yes	No	No
Gershenson	Yes	Yes	No	No	No	Yes	No
Lubras	Yes	Yes	No	No	Yes	No	Yes
Rolland	Yes	Yes	No	Yes	No	Yes	Yes
Somé	Yes	Yes	No	No	Yes	No	No
Easterbrook	Yes	Yes	No	No	No	No	No
Lee	Yes	Yes	Yes	No	Yes	Yes	Yes
Laguna	Yes	Yes	No	No	No	No	Yes
Ramesh	Yes	Yes	Yes	Yes	Yes	No	Yes
Clarkson	Yes	Yes	No	Yes	Yes	No	Yes

1.3 Review of requirements modeling software

Having discussed the capabilities of various requirements modeling approaches in the previous sub-section, this sub-section discusses the capabilities of various requirements modeling software.

UGS Teamcenter [17] focuses on the link between the requirements, mission needs, system objectives, and decisions. Requirement traceability, design allocation and logical verification can all be performed effectively using this model. However, it does not incorporate functionality, testing, and their relation with requirements.

In the V-model of systems engineering, requirements and testing are integral to the development process [18]. Thus traceability of requirements in the V-model is relatively transparent. However the V-model representation does not explicitly show either the components themselves, or the functions they accomplish. It primarily focuses on relations the requirements to tests. The V-model of design is used by DOORS, an object-oriented requirements management software package developed by Telelogic [19].

SysML, the requirements management software package developed by IBM [20], is based on the Unified Modeling Language (UML) architecture used for software engineering. SysML recognizes six top-level categories: structure, behavior, properties, requirements, verification, and propagation. A requirement can include parameters such as an identification tag, the verification method, risk involved, the source of the requirement, and the type of the requirement. Operations performed using the requirements in SysML are allocate, analyze, bind, satisfy, synthesize, trace, and verify.

Enterprise Architect (EA), a Unified Modeling Language (UML) case tool developed by Sparx systems [21], provides the ability to create and view requirements. It also allows the users to enter the attributes of each requirement thereby establishing a systematic approach for sorting the requirements. EA also has a requirements traceability matrix to monitor the impact of change in requirements. However, EA lacks the potential to capture the functions that relate to the requirements or those relating to the components of the system that perform the function.

The AP233 software tool developed by Eurostep facilitates managing, structuring, and allocating requirements as part of systems engineering [22]. This software allows representation at a system's physical and functional levels. Requirements are entered as text and assigned mathematical properties. Traceability is shown via a graphical representation scheme. Although not yet a viable module, current research focuses on implementing algorithms to show derived requirements and to test if requirements

are verified for a given quantitative data. While promising, this tool is not yet fully developed and hence was not available for use in this work.
The capabilities of various requirements management software that are discussed in this section are summarized in Table 2.

Table 2. Comparison of requirements modeling software

Software	Capabilities							
	Req. Allocation	Req. Traceability	Req. Analysis	Configuration management	Change management	Req. Verification	Documentation	Compatibility
SysML	Represented in sparse matrices	Via Requirements Traceability Matrix	Using equations, charts, graphs, circuit diagrams, etc.	Via context blocks	Via stereotype that keeps track of changes	Methods including analysis, test, inspection, demonstration, and test cases	Uses Unified Modeling Language documentation as backend	Microsoft Word
DOORS	(No information available)	Object links provide user defined, multilevel traceability	A set of states are associated with each requirement	Via project baselines	Notifies team members of changes by email	Via automatically generated web interface	Doc-Express add-on module	MS Word and Excel, Telelogic tools and third-party
SLATE	User may allocate portions of a requirement	Uses trace tables	Coupled module	Allows user to standardize all the objects in the database	Accessible parameters	Verification status can be monitored	Uses Frame maker or MS Word	OLE
Enterprise Architect 6.0	(No information available)	Using Relationship Matrix	Requirement traceability is analyzed sequentially	Via color coding	Not supported	(No information available)	Report generator tool	(No information available)

From the literature review on various requirements modeling approaches and requirements management software, it was determined that no single existing modeling approach or software package provided the necessary flexibility to handle the types of requirements and operations that allows for the:

- Creation of models linking requirements, functions, working principles, components and tests
- Examination of the effects of changing any of these entities on the others
- Determination of what effect changes in the model would have on the system performance

2 MODELING SCHEME

2.1 Overview

Our proposed modeling scheme captures how requirements are translated into functions, which are then accomplished by working principles, which are instantiated by physical components, with component parameter values that must be selected. The components as designed must be tested to insure that the final system satisfies the requirements. Individual test measures are correlated with the component parameters that impact them.
To enhance the traceability characteristic of the model, it may be better to adopt conventional approaches like the matrix-based approach, node-link diagrams or connectivity graphs rather than the more abstract linkage in the V-model, for example. Matrix based representations of engineering data are quite common in the literature and facilitate a graphical view of the data captured. The House of Quality (HOQ) [2], the Design Structure Matrix (DSM) [23], Value Analysis [cf., 24], Clarkson et al.'s Change Propagation Method (CPM) [14], and Galvao and Sato's Function Task Interaction Matrix [25] all use matrices to capture multiple domains and facilitate easy visualization. In addition, Keller et al. [26] conducted several usability studies that showed matrix-based representations function as a better visualization tool for analyzing larger spaces. Therefore, for our modeling scheme a matrix based approach is utilized. The model matrices are shown in Figure 1.

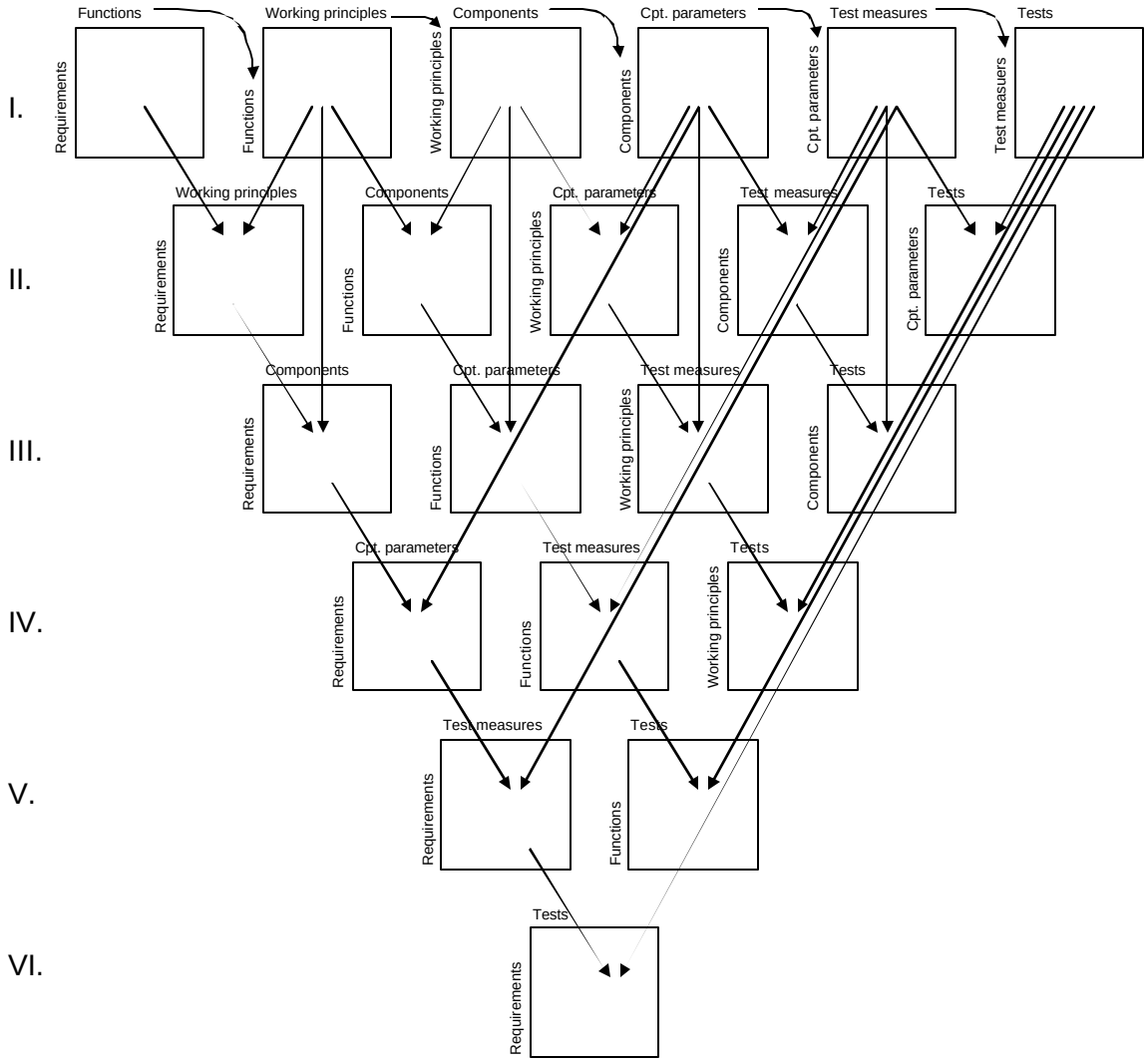

Figure 1. Model matrices capturing seven domains of interest

In order to instantiate the model for any given artifact, only the level I matrices need be populated manually. The level **I**, III, IV, V, and VI levels can be either populated manually or generated automatically through matrix multiplication. For example, the Requirements to Components matrix is generated by multiplying the Requirements to Functions matrix with the Functions to Working Principles matrix with the Working Principles to Components matrix. We have developed a model template automating the matrix multiplication within Microsoft Excel. Similar inter-domain matrix multiplication has been used recently by other authors [e.g., 27, 28]. Example matrices are shown in the case study in Section 3.

2.2 Model consistency

Our modeling framework also incorporates means for consistency checks. Because any matrix below level I may be populated manually, manually populated matrices can be compared to the results obtained through multiplying the higher level matrices. Four categories of comparisons are currently supported within our model template: true positives, false positives, true negatives, and false negatives. A *true positive* indicates a relationship between entities exists in both the manually and automatically populated matrices. A *false positive* indicates a relationship between entities that occurs in the automatically populated matrix, but not in the manually populated matrix. This can be interpreted as either an error in the higher level matrices, or as a relationship that was missed in the manually populated matrix. A *true negative* indicates a relationship between entities does not exist in

both the manually and automatically populated matrices. A *false negative* indicates a relationship between entities that occurs in the manually populated matrix but does not occur in the automatically populated matrix. This can be interpreted as either an error in the higher level matrices, or as a relationship that was erroneously captured in the manually populated matrix.

So far our application of the modeling framework to several automotive assemblies yields roughly 75% combined true positive and true negative and 25% combined false positive and false negative across all multiplied matrices. As discussed in Section 3, there is a trade-off between modeling effort and consistency across the model. A larger, more granular model tends to increase consistency.

Another advantage of capturing the relationships between the domains of interest is the ability to generate intra-domain relationship matrices, similar to Design Structure Matrices (DSMs) [23]. This is accomplished by multiplying the transpose of any matrix in the model with the original matrix. In this fashion, a DSM like matrix is generated for each domain, mapping through every other domain, yielding six automatically generated DSM like matrices for each domain. For example, a component to components can be mapped through requirements, or through functions, or through component parameters, or through tests, or through test measures. These automatically generated matrices can again be compared with manually populated intra-domain relationship matrices to yield another kind of consistency check.

2.3 Numbering scheme

As with any matrix based model, the relationships must be captured according to a predefined scheme. Various schemes have been used in the literature, including binary (0,1), logarithmic (1,3,9), decimal (1-10), and even qualitative schemes such as arrows or partially filled circles. Lewis and Olewnik recently performed a study comparing HoQ matrices populated with several variations of numbering schemes and found decision making based on any of these schemes to be irrational, i.e., no better than decisions based on random numbers [29]. However, our model attempts only to capture and trace relationships, and is not intended explicitly for decision making.

We performed several experiments comparing matrices populated with binary, logarithmic, and decimal numbering schemes. We did not test qualitative schemes because they do not allow for matrix multiplication. We found that the binary scheme was sufficient to capture the relationships involved between domains. Further, we observed that the more granular numbering schemes did not produce significantly better understanding of the relationships between domains, but did require significantly more time to populate the matrices. This was because it is faster and more transparent to determine whether a relationship exists or not, than to then proceed to quantify the strength of that relationship based on scant evidence of each particular relationship. Based on this study, we recommend using a binary numbering scheme with our modeling framework.

Another drawback of using more granular numbering schemes is that the scheme is quickly violated after matrix multiplication. For example, multiplying 1 by 1 still yields 1, whereas multiplying 9 by 9 yields 81, far outside of a (1,3,9) scheme. However, multiplying non-diagonal matrices of even binary elements in general yields entries larger than unity. For example, multiplying a requirements to function matrix with a function to working principles matrix to yield a requirements to working principles matrix yields a matrix where each cell shows the number of functions through which each requirement is related to each working principle; in general the number of functions need not be one.

In a separate study, we were able to recover initial numbering schemes from multiplied matrices. Beginning with a binary numbering scheme, multiplied matrices can be converted to binary simply by replacing every non-zero entry with one. However doing so loses information, as the magnitude of the number in a multiplied matrix does capture in some sense the strength of the interaction between the elements in the underlying matrices. That strength value can be captured and scaled using a more granular numbering scheme such as (1,3,9). We have found that scaling against the largest value that occurs in the matrix is an easy method for interpreting the strengths of the other relationships.

2.4 Hierarchy

The entities in each domain may be organized hierarchically or not, at the discretion of the designer. A hierarchical representation has the advantages of linking entities in different domains that exist on the same level of hierarchy. For example, high level requirements with the high level system, lower level requirements with assemblies, and lowest level requirements with components. Thereby changes to the

system can be studied to the system at different levels of the hierarchy. A change at any level of the hierarchy will propagate down to all lower levels of the hierarchy.
It is important to note, however, that the modeling scheme only captures existence relationships, and does not explicitly capture the designer's knowledge about each entity. For example, in changing the fuel of a vehicle from gasoline to hydrogen, the model would show that an emissions test is linked to the gasoline fuel, however the model would not suggest that a qualitatively different test is needed for hydrogen because of the different chemical properties of hydrogen versus gasoline. Such information it outside of the scope of the model and must be retained by the designer in order to interact intelligently with the model.

3 ANALYSIS METHODS

Analysis of each of the matrices in the model yields information that can be leveraged to improve existing designs and lead to engineering innovation. The matrix based formulation allows for changes to be tracked graphically through the model. For example, the deletion of a requirement can be studied for its effects on components that may no longer be needed, component parameter values that may change, and associated tests that may no longer be needed. Capturing the working principles underlying the component choices also allows for the examination of alternate components that accomplish the same requirement, function, and working principle, only with less cost, weight, environmental impact, or some other criterion. Different working principles can also be explored to satisfy the same requirements and functions, with different working principles leading to different components. Individual matrices facilitate particular analyses as follows (note that not every matrix seems to provide a unique analysis perspective):

- *Requirements to functions*: determine which requirements are functional vs. non-functional
- *Functions to working principles*: explore different working principles to satisfy existing functions
- *Working principles to components*: explore different components for existing working principles
- *Component parameters to test measures*: study effects of optimizing component parameter settings versus their impact on test measures
- *Functions to components*: study how much functionality is associated with each component, identify functional modules, explore strategies for functional integration, identify low functionality components
- *Requirements to components*: determine how much of the system is related to satisfying each requirement, explore effects of removing or adding requirements on the system components
- *Components to tests*: determine which components are related to each test, which components are untested, possible effects of removing or adding tests
- *Requirements to tests*: study of the consistency of the whole model, explore which requirements are tested and untested, see how many relationships map from each requirement to each test

Row sums and column sums can also be used to show the number of relationships related to each entity, which then can be used as a basis for sorting the rows and columns. In the next section we briefly illustrate the modeling scheme and the above analysis methods using reverse engineering data from an existing automotive sub-system.

4 DRIVER'S SEAT EXAMPLE

A modern driver's seat is an electromechanical device that allows for the driver to sit comfortably by adjusting the seat's configuration. The seat is also subject to numerous safety requirements, durability issues, aesthetics, assembly issues, and interfaces with the vehicle frame, seatbelt, electronic controls, and airbags. We gathered seat requirements and tests / test measures, tore-down an existing driver's seat, and described the functionality, working principles, components and component parameters of the seat. All this information was then put into the matrix based modeling scheme. An excerpt from the requirements to functions matrix is shown in Figure 2. The actual number of requirements captured in our analysis is 40, and the actual number of functions captured is 34.

Requirements x Functions	Support legs	Allow adjustment	Absorb energy	deform elastically	protect user from bumps	transfer load from seat springs	allow space for rear passenger's feet	allow forward and rearward adjustment and locking	activate forward and rearward adjustment	Sum
Limit forward seat travel	0	0	0	0	0	0	0	1	1	2
Limit rearward seat travel	0	0	0	0	0	0	0	1	1	2
Allow for user to adjust forward limit	1	0	0	0	0	0	0	1	1	3
Allow for user to adjust rearward limit	1	0	0	0	0	0	0	1	1	3
Visible mechanical areas must be covered	0	0	0	0	0	0	0	0	0	0
Allow user to adjust seat height within target range	0	1	0	0	0	0	0	0	0	1
Allow user to adjust seat angle within target range	0	1	0	0	0	0	0	0	0	1
Allow user to adjust seat depth within target range	0	1	0	0	0	0	0	0	0	1
Allow user to adjust backrest angle within target range	0	1	0	0	0	0	0	0	0	1
Synchronous locking on both sides for forward and rearward adjustment	0	0	0	0	0	0	1	0	0	1
Sum	2	4	0	0	0	0	1	4	4	15

Figure 2. Requirements to functions matrix for driver's seat

This example utilizes our matrix template as implemented in Microsoft Excel, which automatically generates the row and column sums and has conditional formatting which displays cells with zero in light yellow and non-zero cells as bright green. It can be seen that the requirement that the visible mechanical areas be covered does not map to any of the functions in the excerpt. If that observation holds for the rest of the functions not shown, then this requirement would be flagged as non-functional. The functions 'absorb energy,' 'deform elastically,' 'protect user from bumps,' and 'transfer load from seat springs' do not map to any of the requirements in the excerpt. If this observation holds for the rest of the requirements not shown, then these functions may not be necessary, as they are not related to any of the requirements; or this could merely indicate an incomplete understanding of the requirements.

An excerpt from the manually populated requirements to components matrix is shown in Figure 3. Again the actual number of requirements captured in our analysis is 40, while the actual number of components captured is 28.

Requirements x Components	Seat	Headrest assembly	Seat backrest assembly	Seat backrest cover	Sum
Seat must fit in defined position in vehicle	1	0	0	0	1
Seat must place driver in defined position in vehicle	1	0	0	0	1
Seat must attach to the vehicle	1	0	0	0	1
Seat must not distort upon assembly	1	0	1	1	3
Allow for grabbing tool handling device	1	0	1	1	3
Allow electronic interface to vehicle	1	0	0	0	1
Limit forward seat travel	1	0	0	0	1
Limit rearward seat travel	1	0	0	0	1
Allow for user to adjust forward limit	1	0	0	0	1
Allow for user to adjust rearward limit	1	0	0	0	1
Visible mechanical areas must be covered	1	1	1	1	4
Sum	11	1	3	3	

Figure 3. Manually populated requirements to components matrix for driver's seat

In this example the requirement on covering the mechanical areas is related to the most components. Conversely, the headrest assembly is related to the least requirements. The components are arranged

hierarchically, therefore the seat is related to all the requirements, the individual assemblies are related to groups of requirements, and the individual components are related to one or more requirements.
An excerpt from the automatically generated requirements to components matrix is shown in Figure 4.

Requirements x Components	Seat	Headrest assembly	Seat backrest assembly	Seat backrest cover	Sum
Seat must fit in defined position in vehicle	0	0	0	0	0
Seat must place driver in defined position in vehicle	0	0	0	0	0
Seat must attach to the vehicle	3	0	1	0	4
Seat must not distort upon assembly	3	0	1	0	4
Allow for grabbing tool handling device	3	0	1	0	4
Allow electronic interface to vehicle	0	0	0	0	0
Limit forward seat travel	2	0	0	0	2
Limit rearward seat travel	2	0	0	0	2
Allow for user to adjust forward limit	3	0	0	0	3
Allow for user to adjust rearward limit	3	0	0	0	3
Visible mechanical areas must be covered	4	1	1	1	7
Sum	23	1	4	1	

Figure 4. Automatically generated requirements to components matrix for driver's seat

Comparing the manually populated with the automatically generated matrices shows that the automatically generated matrix does capture the additional information in the magnitudes of the numbers, which reflect the number of functions and working principles through which each requirement is related to each working principle. Obviously the magnitudes are higher the higher each component is in the hierarchy. For example the relationships for the seat are higher in magnitude than for any of the assemblies or individual components.
Analyzing the consistency between the complete manually populated and the complete automatically generated matrices (not shown), 12% of the values are true positives, 85% of the values are true negatives, 0% of the values are false positives, and 3% of the values are false negatives. Combining the true positives with true negatives yields an overall consistency metric of 97%, which is higher than most matrices we have studied. In the excerpts shown in Figures 2 and 3, the cells that map the requirement 'Visible mechanical areas must be covered' to all four components shown are examples of *true positives*. The lack of relationship shown between the headrest assembly and the first requirement ('Seat must fit in defined position in vehicle') in both matrices is an example of a *true negative*.
The relationship between the requirement 'seat must attach to the vehicle' and the 'seat backrest assembly' which appears in the automatically generated matrix, but not in the manually populated matrix, is an example of a *false positive*. In this case the seat backrest assembly does not attach to the vehicle; only the seat base does that. The appearance of the false positive is due a lack of sufficient granularity in the list of functions. The requirement 'seat must attach to the vehicle' is mapped through the function 'transfer load' which then maps through the common working principle of a 'space frame' to both the seat base and the seat backrest. Had the functionality of transfer load been decomposed further to account for the different loads the backrest and the seat base experience, this modeling false positive could have been avoided. This points to the fact that there is a trade-off between model granularity and model consistency. Some inconsistencies in the model may be allowable in order to keep the modeling effort reasonable.
The relationship between the seat and the first requirement, which does appear in the manually populated matrix, but does not appear in the automatically generated matrix, is an example of a *false negative*. In this case the lack of this relationship in the automatically generated matrix can be traced back to the fact that 'Seat must fit in defined position in vehicle' is not a functional requirement (rather it is a constraint on the geometry) and so does not map to any functions in the requirements to function matrix, and thus the relationship does not appear through multiplication of the requirements to function matrix in generated the requirements to components matrix.

Figure 5 shows all six possible intra-domain requirements to requirements matrices for the driver's seat. The figures are intended only to show the trends across the six different configurations, and as the matrices show all 40 requirements, however the headings are not shown for clarity. Similar clusters are evident across the six matrices. Note that only the requirements to requirements matrix through functions (Figure 5a) is based on a Level I matrix, whereas the other five matrices are generated using the automatically generated lower level matrices. Compared to the matrix shown in Figure 5a, the other five matrices display similar true positives and true negatives, while showing varying amounts of false positives and false negatives. The overall trend appears to match the 75% consistency measure found in the automatically generated Level II-VI matrices.

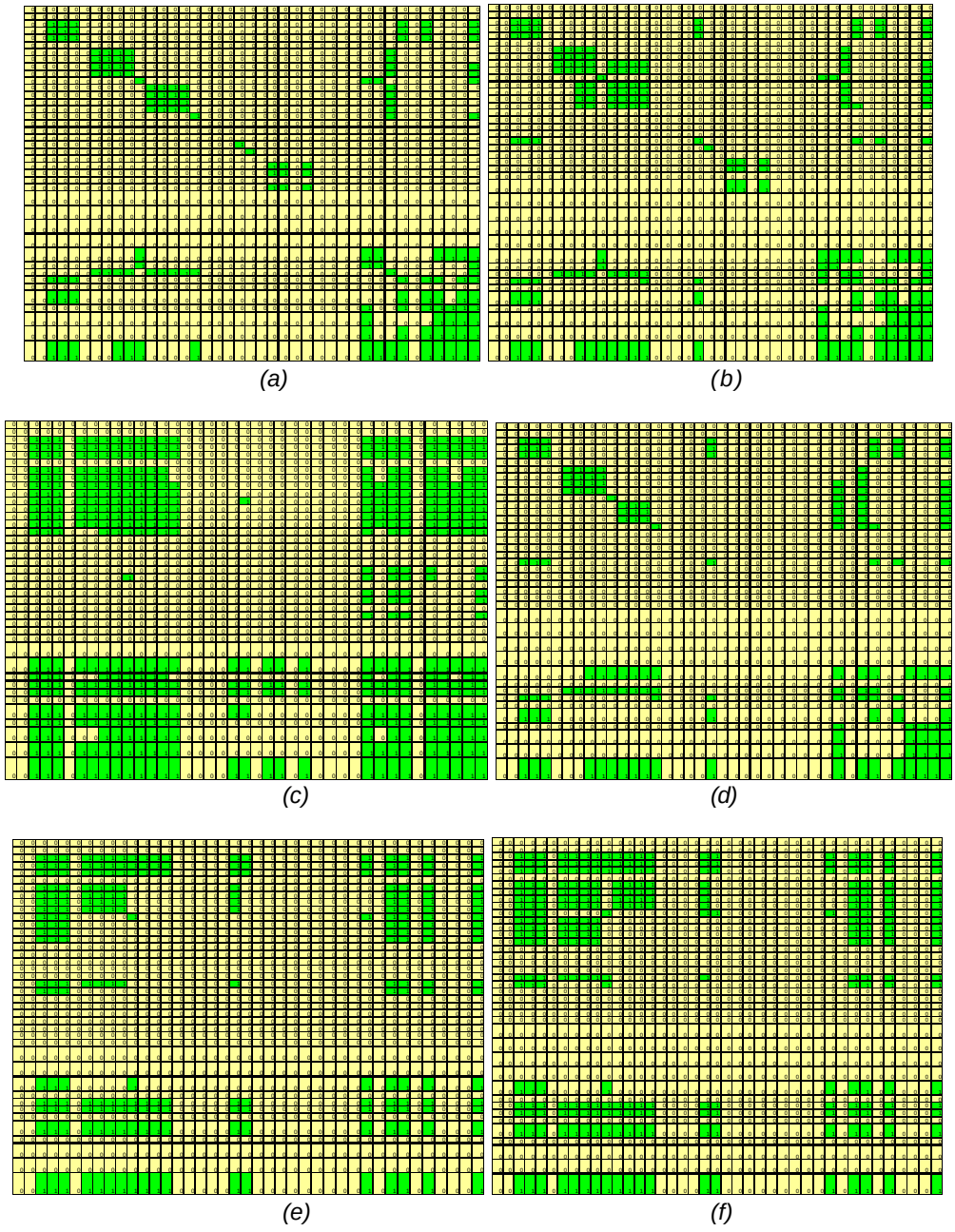

Figure 5. Requirements to requirements matrices through (a) functions, (b) working principles, (c) components, (d) component parameters, (e) tests and (f) test measures

4. SUMMARY REMARKS

In this paper we have introduced a hierarchical modeling scheme developed to support engineering innovation. Our model presents what we believe is a more complete framework for capturing and analyzing the information needed for a systematic design process, and captures more detail and facilitates more analyses than other matrix based tools. However the matrix format presents an easy to understand and easy to populate model that engineers can interact with, similar to other matrix based tools commonly used in industry.

REFERENCES

[1] Pahl G. and Beitz W. *Engineering Design: A Systematic Approach 2nd ed*, 1996 (Springer, New York).

[2] Hauser J.R. and Clausing D. The House of Quality. *Harvard Business Review*, 1988, 3, pp. 63-73.

[3] Mocko M., Summers J.D., Fadel G.M., Teegavarpu S., Maier J.R.A. and Ezhilan T. A modeling scheme for capturing and analzyzing multi-domain design information: a hair dryer example, *International Conference on Engineering Design, ICED '07*, Paper no. 897. Paris, France, 2007 (The Design Society).

[4] Pugh S. *Total Design, Integrated methods for successful product engineering*, 1991 (Addison Wesley)

[5] Lubars M., Potts C. and Richter C. A review of the state of the practice in requirements modeling, in *Proceedings of the International Requirements Engineering Symposium*. Los Alamitos, CA, 1992: IEEE Computer Society Press, pp. 2-14.

[6] Easterbrook S., Lutz R., Covington R., Kelly J. , Ampo Y. and Hamilton D. Experiences using lightweight formal methods for requirements modeling, *IEEE Transactions on Software Engineering*, 1998, 24(1):4-14.

[7] Lee J. and Kuo J.Y. New Approach to Requirements Trade-Off Analysis for Complex Systems, *IEEE Trans. Knowledge and Data Eng.*, 10(4).

[8] Gershenson J.K. and Stauffer L.A. A Taxonomy for Design Requirements from Corporate Customers. *Journal of Research in Engineering Design*, 1999 11(2): 103-115.

[9] Rolland C. and Prakash N. From conceptual modelling to requirements engineering. *Annals of Software Engineering*, 2000 10: 151-176.

[10] Ramesh B. and Jarke M. Toward Reference Models for Requirements Traceability. *IEEE transactions on Software Engineering*, 2001 27(1).

[11] Laguna M. et al. Open Issues in Requirements Modeling for Reuse, in *Applying Requirements Engineering*, Durán, A, and M. Toro, eds., Salamanca, Spain, 2002 (Catedral Publ), pp. 73-88.

[12] Delugach H. S. A Multiple Viewed Approach to Software Requirements, *Ph.D. dissertation*, Dept. of Computer Science, University of Virginia, 1991.

[13] Fu M.W. and Lu W.F. Modeling and Management of Design Requirements in the Product Development Life Cycle", *ASME Design Engineering Technical Conference and Computer and Information Engineering Conference*, Chicago, Illinois, USA, 2-6 September, 2003 (ASME).

[14] Clarkson P.J., Simons C., Eckert C.M. Predicting change propagation in complex design. *ASME Journal of Mechanical Design* 2004, 126: 765-797.

[15] Somé S. Supporting use case based requirements engineering, *Information and Software Technology*, 2005.

[16] Hull E., Jackson K. and Dick J. *Requirements Engineering*, 2004 (Springer, New York).

[17] UGS Teamcenter software. http://www.ugs.com/products/teamcenter/ accessed 7 February 2007.

[18] Dick J. (2004) "What is Requirements Management?" White paper, Version 1, 05 November 2004 (Telelogic). http://download.telelogic.com/download/paper/WPWhatIsRMNov04.pdf

[19] Telelogic DOORS software. http://www.telelogic.com/products/doors/doors/index.cfm accessed 7 February 2007.

[20] Conrad B., *SysML and UML 2 Support for Activity Modeling*. (Wiley InterScience) DOI 10.1002/sys.20046.

[21] Sparx Systems. Requirements Management with Enterprise Architect. http://www.sparxsystems.com/downloads/whitepapers/Requirements_Management_in_Enterprise_Architect.pdf accesssed 7 February 2007.
[22] Eurostep Group AB. A beginner's guide to AP233. http://ap233.eurostep.com/Beginners/Beginners%20Guide%20to%20AP233_files/frame.htm accessed 7 February 2007.
[23] Browning T .R. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions, *IEEE transactions on Engineering Management*, Vol48, No.3, August 2001
[24] Otto K. and Wood K. *Product Design*, 2001 (Prentice-Hall, Upper Saddle River, NJ).
[25] Galvao, A. B. and Sato K. Affordances in Product Architecture: Linking Technical Functions and User Requirements, *Proceedings of ASME Design Theory and Methodology Conference*, Long Beach, CA, 2005. Paper no. DETC2005-84525.
[26] Keller R, Eckert CM, Clarkson PJ, Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? Information Visualization (2006) 5, 62–76. doi:10.1057/palgrave.ivs.9500116
[27] Leung P., Ishii K., Benson J. and Abell J. System Engineering Workshare Risk Analysis. In *11th ASME Design for Manufacturing and the Life Cycle (DFMLC) Conference*, Philadelphia, Pennsylvania USA, 2006, Paper No. DETC2006-99252.
[28] Johannson O. and Krus P. Configurable Design Matrices for Systems Engineering Applications. In *26th ASME Computers and Information in Engineering (CIE) Conference*, Philadelphia, Pennsylvania USA, September 2006, Paper No. DETC2006-99481.
[29] Olewnick, A. and K. Lewis. Can a House Without a Foundation Support Design? *Proceedings of ASME Design Theory and Methodology Conference*, Long Beach, CA, 2005. Paper No. DETC2005-84765.

Contact: Georges Fadel
Clemson University
Department of Mechanical Engineering
202 Fluor Daniel EIB
Clemson, SC 29634-0921
USA
Phone: 864-656-5620
Fax: 864-656-4435
e-mail: fgeorge@clemson.edu
URL: http://www.ces.clemson.edu/me/credo