SysCon 2008 – IEEE International Systems Conference
Montreal, Canada, April 7–10, 2008

Using Attribute Classes to Uncover Latent Value during Conceptual Systems Design

Adam M. Ross and Donna H. Rhodes
Engineering Systems Division, Massachusetts Institute of Technology
Building NE20-388, 77 Massachusetts Ave
Cambridge, MA 02139
http://seari.mit.edu

Abstract - *A key challenge for designers is to create systems that stakeholders will perceive as delivering sustained value over the life of the system. The perceived value of a system by its stakeholders changes over time as a result of many different factors such as experience with use of the system, changes within the regulatory environment or marketplace, availability of new technologies, and emergent needs. During the concept phase, designers elicit stakeholder needs and desired system attributes though various methods, yet there is often significant unarticulated or latent value that remains uncovered until later in the system lifecycle. A method is presented that uses attribute classes to aid the system designer in understanding perceived value in context of an overall value spectrum. Desired system attributes are characterized using several value classes including: articulated value, free latent value, combinatorial latent value, accessible value and inaccessible value. Illustrative examples are presented to examine how this method can aid the designer in a deeper exploration of system attributes to uncover latent value during the conceptual system design phase. Implications of this method for improving the overall system design process are discussed, including strategies for bearing the costs of latent value.*

Keywords - *system design, articulation of value, latent value, attribute classes.*

I. Introduction

One problem that faces all system designers is that of creating value by having the experiences with a system meet the expectations of system stakeholders. It is the creation of value that motivates the design effort, without which, systems face failure and developers face the consequences of that failure. Dynamic contexts due to environment changes, new expectations of stakeholders as they learn or are influenced by others, and introduction of new technologies and new competitors, can significantly affect the perceived success for a system. Instead of resisting the inevitable change in stakeholder value expectations, system designers can proactively embrace the possibilities of change by building into the system the ability to provide future value. Using *attributes* as decision metrics for differentiating the "goodness" of alternatives, the concept of attribute classes is introduced as a framework for thinking about actual and potential value perception by stakeholders.

The distinguishing characteristic that determines an attribute classification is the cost to "display" or "activate" an attribute when a stakeholder desires to see such an attribute. Unarticulated value, that which is not explicitly communicated, perhaps because it is unrecognized, can be explicitly managed through the attribute classification system by increasing the potential for a system to meet needs as they become expressed.[1] As the cost to redesign a system increases, the importance increases for a designer to be able to anticipate and design in latent value that will increase the likelihood of sustaining system success through continued perception of delivering value to stakeholders. The ultimate goal of design using attribute classes is to be able to match dynamic system characteristics to dynamic value expectations.

II. Value Defined

Value can be defined as relative worth, utility, or importance; it is the quality of a thing considered in respect to its power and validity for a specified purpose or effect. The concept of value is at once abstract and yet pervasively accessible. The pursuit of value motivates exchange in markets, both formal and informal, as well as impacting the discipline of design. Due to its inherently subjective nature, perceptions of value must be effectively communicated between value-consuming and value-creating entities in order to ensure needs are met. The communication, or articulation, of value is a core concept in the design process, often represented as "needs identification" in traditional development processes [1].

The notion of articulated value is discussed in [2,3] and for the purposes of this paper will be assumed to include the explicitly communicated desires, or elicited attribute set, for each decision maker. The unexpressed, or unarticulated, values include those "somethings" that give value to a given decision maker, but for one reason or another were not elicited in the attribute set. Reasons for decision makers having unarticulated values range from "could not" to "would not" to "should not" say, and can sometimes occur when values are assumed to already be known.

[1] For the purposes of this paper, *unarticulated value* is taken to mean those form or function aspects of a design that provide utility to a stakeholder, but which are not explicitly communicated by the stakeholder to the designer. Unarticulated values may or may not become articulated over time. *Unarticulated value* in social science and management often refers to the unexpressed framework that provides the context for judgment. While the meaning in this paper is related, it is more narrowly taken to consider specific aspects of a design, rather than a frame of reference for judgment.

Ref. [4] discusses the approach called "value-focused thinking" as opposed to "alternatives-focused thinking." The key differentiator between these approaches is to recognize that understanding the core value propositions of a decision maker can enable decision opportunities, especially to create new or additional value, as opposed to trying to seek criteria and justification for already offered alternatives.

III. Eliciting Value

The typical approach to value elicitation is through the traditional requirements engineering elicitation approach [5]. Techniques for requirements elicitation may include interviews, focus groups, Delphi technique, and soft systems methodology [6]. The strength of this approach is that it uses structured processes and enabling techniques to identify and capture stakeholder needs and preferences. Some of the weaknesses that have been cited with this approach include: (1) stakeholders have incomplete understanding of their needs; (2) system users and the needs elicitation analyst speak difference languages; (3) boundaries of the system are ill defined; (4) unnecessary and confusing design or implementation information may be given instead of identifying the core need; and (5) it is often the case that "obvious" information is omitted and thought by the stakeholder to be assumed by the designer [7].

The attention to value elicitation has a long history. Value Engineering has been used for over 50 years, and is "an organized/systematic approach directed at analyzing the function of systems, equipment, facilities, services, and supplies for the purpose of achieving their essential function at the lowest life-cycle cost consistent with required performance, reliability, quality, and safety" [8]. The focus is on attainment of return on investment by improving what the system does in relation to money spent. Value engineering offers sound processes and philosophies for the overall consideration of value throughout the program lifecycle.

Another approach used in eliciting value is in the definition of utility functions in terms of attribute weights and scores through many different techniques [9]. Techniques include direct elicitation, ranking methods, indifference methods, and assessment through direct observation [10].

IV. Perceived Value Spectrum

In the dialogue between the decision maker and a stakeholder, the designer seeks to identify the criteria for maximal value through the articulation of objectives, requirements, and attributes. Unarticulated value may remain within the overall value spectrum, as shown in Figure 1, given aspects that decision makers can't say, don't say, or won't say during the elicitation process.

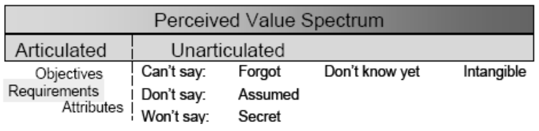


Fig. 1. Perceived Value Spectrum

The perceived value spectrum can be used to remind a designer of possible types of value to consider beyond those articulated by stakeholders. Elucidators, or mechanisms to move the values from unarticulated to articulated perceived-value categories, include the following: personal reflection (time); conversations using mediators (facilitation); experience with the system (learning by doing); interactions with system context (competition, test-driving); and change in context (change of the "rules").

A key question confronting the designer at this point is how to deliver value in the face of various value-perceptions. One process, utilized extensively in economic market analyses is that of revealed preferences. Revealed preferences are preferences captured through statistical analysis of observed choices of decision makers. Presumably decision makers choose systems that deliver value and thus reveal information about their preference structure, or tastes.

In certain situations, however, such data is unavailable due to the limited or specialized nature of a system, as well as a limited number of purchase or acquisition decisions. The unique context of each system need may entail a specialized set of preferences that cannot be garnered from past behavior. Instead of relying on statistical analysis of past behavior, revealed preferences could be captured through conversations with decision makers about hypothetical system choices in the current context. Unfortunately, the process of preference elicitation typically does not give a complete picture to the designer. Additionally, conflicting preferences of multiple decision makers may be revealed that do not point the designer to an obvious aggregate preference set for maximizing delivered value. Complicating matters further are apparent dynamic preferences.

The causes of apparent dynamic preferences of a decision maker to the designer include: (1) personal drift of the decision maker's thinking; (2) changing context affecting the dilemma being considered by the decision maker; and (3) the movement of needs from unarticulated to articulated. Whatever the case, in order to maximize delivery of value, the designer must match a system to the dynamic current preferences to the best extent possible. Since the decision maker's personal drift is the most difficult to ascertain, attention to the context as well as to the needs articulation should be a significant focus of the designer.

V. Value Metrics: Attributes

The stakeholder role represents those individuals, groups, and entities who derive value from association with the system. Stakeholders in general, however, often have little direct influence over the creation of the system itself. If the goal of the designer were to maximize value delivered to the entire stakeholder set, some method for capturing each stakeholder's value proposition would be necessary in order to have a direct effect. Even if such an undertaking were possible, the picture would still be incomplete. In addition to need, a system requires resources. Resources are the raw and mediating materials, processes, and expertise, both tangible and intangible, which are used to create the system. The gatekeepers for both the need and resources are the decision makers, a subset of stakeholders with significant influence over either the driving need or resource allocation that affects system creation. Since the decision maker(s) wield(s) the power over whether a system is created, the designer should focus on maximizing value to the decision maker(s).[2] Each decision maker has a set of objectives about which decisions are made.

Attribute-based value is an effort to operationalize the concept of objective-driven decision-making. The following is a question to pose to decision makers when eliciting attributes: "when making a decision about a particular option, what are the characteristics that you would look at for differentiating the 'goodness' of alternatives?" Those characteristics are the attributes. An attribute is a decision maker-perceived metric that measures how well a decision maker-defined objective is met. For each attribute, the following must be defined: definition in context, units, and range from least to most acceptable levels. The definition should be developed in concert with the decision maker in order to ensure the decision maker actually has value perception over it. The range reflects the fact that value is perceived for multiple attribute levels, and in the limit the range converges to a point, the attribute becomes a requirement. Of course a decision maker could have multiple objectives and therefore a set of attributes. According to [11], an attribute set must be complete, operational, decomposable, non-redundant, minimal, and perceived independent. Operational means that the decision maker actually has preferences over the attributes. Decomposable means that they can be quantified. Non-redundant means none are double-counted. Minimal and complete are in tension, since a designer seeks to capture as many of the important decision metrics as possible, while keeping in mind human cognitive limitations. In practice, no set can be guaranteed to have all of these properties. The problem of completeness applies just as easily in the requirement generation process in standard engineering practice. Designers must do the best they can.

VI. Sources of Value: Attribute Classes

The elicitation of attributes, both articulated and unarticulated, can be done through a facilitation process mediated by system designers. The literature on requirements generation can inform the elicitation process. Section 3.2, "Preference Capture," of [2] describes the general concept of attribute elicitation for use within Multi-Attribute Utility Analysis and Prospect Theory, two decision-analytic theories that can be used to improve engineering design decision-making [4,12]. Putting attributes into a temporal discovery context, [13] provides a framework for thinking about the evolution of articulated needs from fuzzy wants through attribute definition down to concrete requirements. Formal interviews, group discussions, learning by doing ("playing" or "test driving" the system), and introspection are just a few of the methods that can be used to elicit value propositions from decision makers.

Throughout elicitation, it is important to keep in mind the concept of "framing." Framing represents the cognitive context from which a decision maker considers a problem. For example the same outcome could be cast in terms of a "cost" or in terms of an "uncompensated loss" and will be perceived differently by the decision maker. Cognitive bias as a result of framing is a well documented phenomenon in the psychology literature. Ref. [14] contains a collection of several dozen such papers, including descriptions of Prospect Theory, a theory of value combining insights of cognitive biases from Psychology into an Economic model of choice. Consistency and care in the framing of attribute elicitation is essential to ensure reliable and repeatable value perceptions. It is important for the analyst to be able to distinguish changes in value perception due to a real underlying value perception change versus errors in measurement due to cognitive biases or inconsistencies in framing for attribute elicitation.

In terms of capturing value propositions, the previously developed concept of attributes can be used as a metric to ascertain how well objectives deliver value. Spanning the range from known articulated value to unknown unarticulated value, attributes can be classified by the designer in terms of potential cost for a system to "display," as shown in Table 1 below.

Table 1. Attribute classification (0 to 4)

Class	Name	Property of Class	Cost to Display
0	Articulated Value	Exist and assessed	0
1	Free Latent Value	Exist, not assessed	0
2	Combinatorial Latent Value	Can exist by recombining class 0 and 1 attributes	Small
3	Accessible Value	Can be added through changing the design variable set (scale or modify system)	Small→large
4	Inaccessible Value	Cannot be added through changing design variable set (system too rigid)	Large→infinite

In order to deliver value, an attribute must be perceived by a decision maker, as well as be "displayed" by the system. The "existence" of an attribute means that the system has either the form or function specified by the attribute and is thus "displayed." "Articulation" refers to explicit communication by a decision maker to the designer that a particular attribute or set of attributes is value-perceived. "Potential" attributes are those that could be "displayed" by the system if the system were changed in some way. For the following attribute class definitions, a state 1 system is the "original" or "as-designed" system, while a state 2 system is the "changed" system. A dynamically successful system is one that can be changed to continually match its displayed attributes to those that are articulated, and thus explicitly expected, by decision makers.

[2] Additionally, designers may also need to consider stakeholders with "veto" power, or the ability to interfere with or stop a development effort.

VII. Attribute Classes

In order to inspire designers to consider sources of value beyond that which is articulated in the upfront needs identification of traditional design processes, five attribute classes are proposed in the following framework for defining system value: articulated value, free latent value, combinatorial latent value, accessible value and inaccessible value. It is intended that the classification framework will allow designers to consider more sources of value and to develop system strategies for lowering the cost to "display" delayed or time-varying articulated attributes. The classes are described and illustrated in the following sections.

A. Class 0 Attributes: Articulated Value

The first attribute class to consider has those that are typically included in traditional design: the articulated attribute set, as shown in Figure 2. The *class 0 attributes* are those explicitly communicated as an expectation by a decision maker and are designed into and "displayed" by the system. The "cost" to add class 0 attributes to a state 2 system is zero since the attributes are already displayed by the state 1 system.

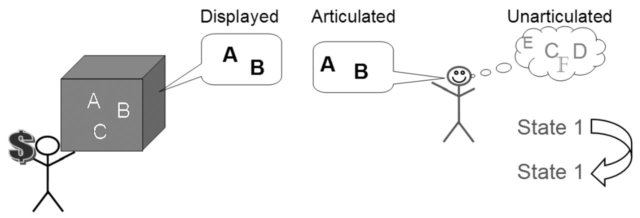


Fig. 2. Class 0 Attributes: Articulated Value

As an example, class 0 attributes are equivalent to the requirements currently met by the system. A cell phone that provides, and was designed to provide, good sound quality and durable construction meets the articulated values of the consumer who explicitly demands such attributes.

B. Class 1 Attributes: Free Latent Value

In addition to displayed attributes that are value-perceived, a system can display a number of other attributes, which are not value-perceived, as shown in Figure 3. The *class 1 attributes* represent a type of latent value. If a decision maker adds such an attribute to his value-perceived set, no "cost" is incurred to change the system since it is already displayed.

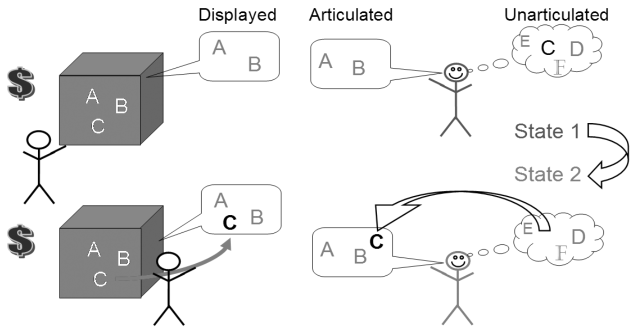


Fig. 3. Class 1 Attributes: Free Latent Value

As an example, consider a customer purchasing a new car. Entering the showroom, the customer may consider body styling and gas efficiency as his decision criteria, or articulated attributes. Upon test driving a few cars, he comes to realize that comfort also generates value and was a previously unarticulated value. Cars being considered already "display" the comfort attribute and thus do not require modification to deliver comfort value to the customer. In this example, comfort is a free latent value for the existing car.

C. Class 2 Attributes: Combinatorial Latent Value

The next class of attributes captures the other type of latent value in a system: that which can be introduced into the system through small cost by recombining existing attributes. The system itself does not require a change, rather the interpretation of the existing attributes may require minor change as shown in Figure 4. The cost of such recombination is much less than that which would be required to change the initial system itself, thus these *class 2 attributes* are "combinatorial latent value."

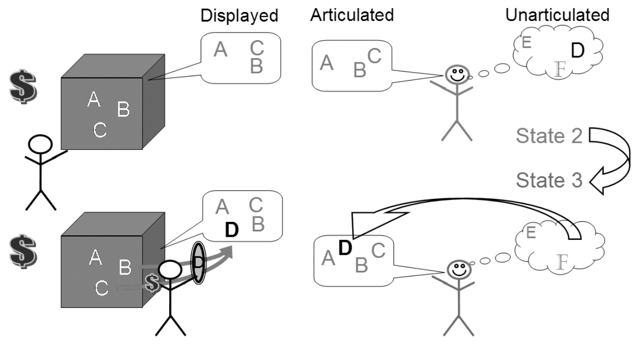


Fig. 4. Class 2 Attributes: Combinatorial Latent Value

As an example, consider the GPS system, with two of its attributes: to provide time, and to provide position data. Suppose initially, the decision maker cares about these two capabilities, which are his attributes. Later, the decision maker realizes that he also cares about his velocity. The system designer wants the system to continue to deliver value. The new attribute, velocity, can be derived from a recombination of existing capabilities. The system itself requires no change, rather a hand held device or other such interpretive system, can be used to derive the new attribute. Another example of combinatorial latent value for GPS is interactive navigation system in cars, providing real time driving directions to destinations of interest. Compared to changing the GPS system, such new capability is very inexpensive.

D. Class 3 Attributes: Accessible Value

When the system itself must be changed in order to "display" a new attribute, such an attribute belongs to *class 3*, if the cost is not unreasonable. The cost of such a change can vary from small to large, and each decision maker subjectively defines the reasonability of that cost, usually informed by resource constraints. Even though the system must be changed, the attributes created in this way are "accessible value" as illustrated in Figure 5.

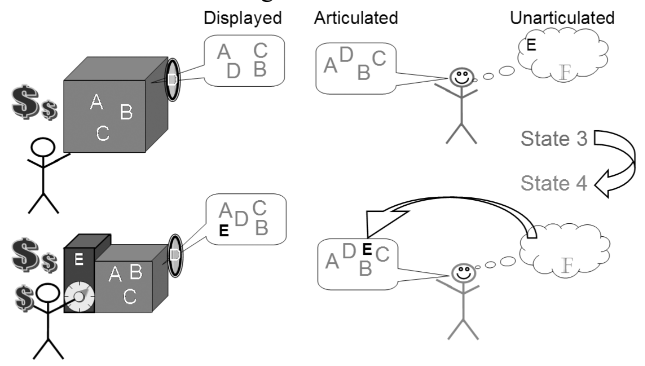


Fig. 5. Class 3 Attributes: Accessible Value

As an example, consider an audiophile consumer with an adequate stereo system. Suppose the consumer wishes to add to the system the ability to play MP3 format audio files. In order to add this capability, the system itself requires modification. Options include replacing the CD player with an MP3-compatible player, or perhaps modification of the current system software to enhance audio decoding. In either case, the system must be changed to display MP3-playing capability.

E. Class 4 Attributes: Inaccessible Value

When the system cannot be changed or the cost incurred is too extreme to enable the system to display a new attribute, such an attribute belongs to *class 4*. These attributes do not flow from the particular system concept being considered, or perhaps represent an unreasonable burden to include, and are "inaccessible value" to the system under consideration.

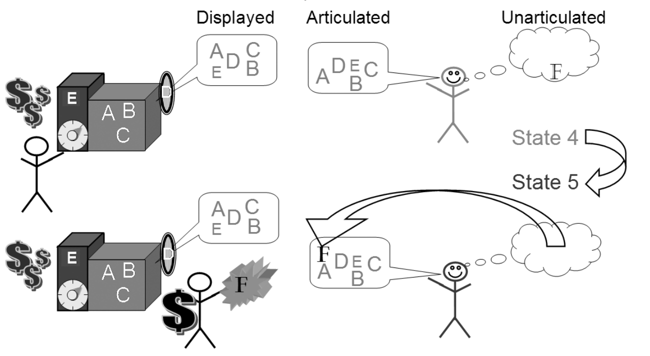


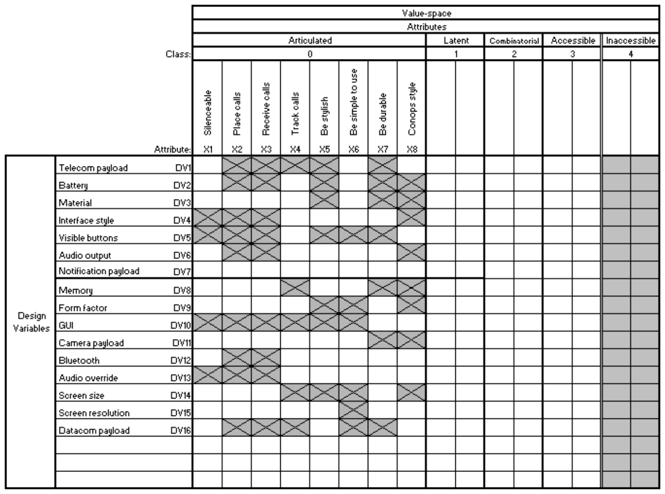
Fig. 6. Class 4 Attributes: Inaccessible Value

As an example, consider the desire to include a food preparation capability into a passenger car. While having that capability might add value to a particular decision maker, the cost of doing so is either prohibitive, or would require the concept of car to be revisited. A camper, however, often does include a kitchen and could readily accommodate such a new attribute, but the transition from passenger car to camper is not inexpensive and requires discontinuity in concept.

VIII. Example Application of Framework

The following is a simplified example of the attribute class framework applied to a cell phone design. The decision maker for this case is a user. For additional example cases, [3] provides examples of the framework applied to the Joint Direct Attack Munition (JDAM) system and the Terrestrial Planet Finder (TPF) system.

First, the attributes of a user are elicited, which define the articulated value expectations of the user. These class 0 attributes can be used to motivate the generation of design variables, as described in [2,3]. Figure 7 demonstrates the example applied to a user: USER-A. Along the top of the Design-Value Matrix (DVM) are listed the explicitly requested form and function attributes of the user [3]. Design variables, which encapsulate designer-controllable features of the cell phone concept, are mapped to the attributes to drive maximization of user perceived value. Some listed design variables are intended to drive class 1 attributes, which follow.

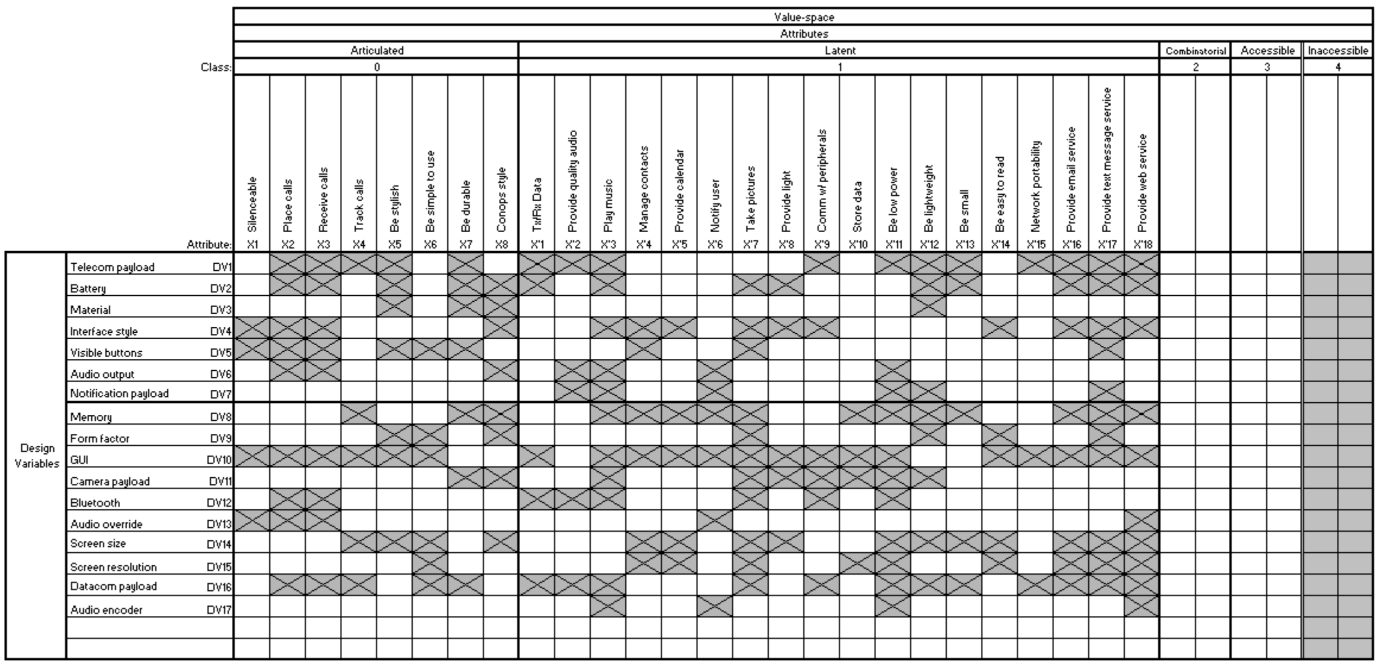


		Value-space															
		Attributes															
		Articulated								Latent		Combinatorial		Accessible		Inaccessible	
	Class:	0								1		2		3		4	
		Silenceable	Place calls	Receive calls	Track calls	Be stylish	Be simple to use	Be durable	Conops style								
	Attribute:	X1	X2	X3	X4	X5	X6	X7	X8								
Design Variables	Telecom payload DV1		X	X	X	X		X									
	Battery DV2		X	X		X		X	X								
	Material DV3					X		X	X								
	Interface style DV4	X	X	X					X								
	Visible buttons DV5	X	X	X		X	X	X									
	Audio output DV6		X	X					X								
	Notification payload DV7																
	Memory DV8				X			X	X								
	Form factor DV9					X	X		X								
	GUI DV10	X	X	X	X	X	X										
	Camera payload DV11							X	X								
	Bluetooth DV12		X	X													
	Audio override DV13	X	X	X													
	Screen size DV14				X	X	X		X								
	Screen resolution DV15						X										
	Datacom payload DV16		X	X	X		X	X									

Fig 7. Design-Value Matrix with class 0 attributes mapped to design variables

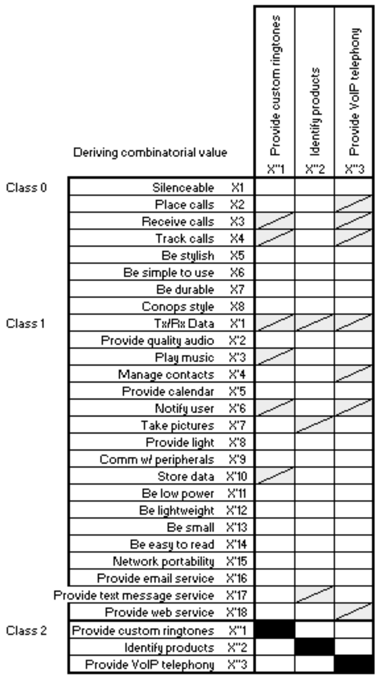
When a system will be developed to satisfy multiple users, attributes articulated by one user, may not be articulated by another user and therefore a single design will have latent value generated by meeting other users' needs. Using the same matrix representation, the designer lists attributes desired by other users as latent attributes, and generates additional design variables to drive those latent attributes. The designer should inspect the design variables and infer additional latent value in the already expressed concept. Figure 8 demonstrates the expanded DVM with class 1 attributes added.

Given that other users may desire a cell phone that can play music or take pictures, the design parameters of "audio encoder" and "camera payload" are added to the design variable set and create latent value for USER-A. If USER-A does not desire these functions, then USER-A will be paying for these unnecessary functions as long as the design incorporates these features.



			Value-space																													
			Attributes																															
			Articulated								Latent																		Combinatorial		Accessible		Inaccessible	
		Class:	0								1																		2		3		4	
			Silenceable	Place calls	Receive calls	Track calls	Be stylish	Be simple to use	Be durable	Conops style	Tx/Rx Data	Provide quality audio	Play music	Manage contacts	Provide calendar	Notify user	Take pictures	Provide light	Comm w/ peripherals	Store data	Be low power	Be lightweight	Be small	Be easy to read	Network portability	Provide email service	Provide text message service	Provide web service						
		Attribute:	X1	X2	X3	X4	X5	X6	X7	X8	X'1	X'2	X'3	X'4	X'5	X'6	X'7	X'8	X'9	X'10	X'11	X'12	X'13	X'14	X'15	X'16	X'17	X'18						
Design Variables	Telecom payload	DV1		☒	☒	☒	☒		☒		☒	☒	☒						☒		☒	☒	☒		☒	☒	☒	☒					▨	▨
	Battery	DV2		☒	☒		☒		☒	☒	☒		☒				☒	☒				☒	☒			☒	☒	☒					▨	▨
	Material	DV3					☒		☒	☒												☒											▨	▨
	Interface style	DV4	☒	☒	☒					☒			☒	☒	☒		☒	☒	☒					☒		☒	☒	☒					▨	▨
	Visible buttons	DV5	☒	☒	☒		☒	☒	☒					☒			☒										☒						▨	▨
	Audio output	DV6		☒	☒					☒		☒	☒			☒					☒												▨	▨
	Notification payload	DV7										☒	☒			☒					☒	☒					☒						▨	▨
	Memory	DV8				☒			☒	☒			☒	☒	☒	☒	☒			☒	☒	☒	☒			☒	☒	☒					▨	▨
	Form factor	DV9					☒	☒		☒							☒					☒		☒			☒						▨	▨
	GUI	DV10	☒	☒	☒	☒	☒	☒			☒		☒	☒	☒	☒	☒	☒	☒	☒	☒			☒	☒	☒	☒	☒					▨	▨
	Camera payload	DV11							☒	☒			☒				☒	☒	☒	☒	☒	☒											▨	▨
	Bluetooth	DV12		☒	☒						☒	☒	☒				☒		☒		☒												▨	▨
	Audio override	DV13	☒	☒	☒											☒												☒					▨	▨
	Screen size	DV14				☒	☒	☒		☒				☒	☒		☒	☒			☒	☒	☒	☒		☒	☒	☒					▨	▨
	Screen resolution	DV15						☒						☒	☒		☒			☒	☒			☒		☒	☒	☒					▨	▨
	Datacom payload	DV16		☒	☒	☒		☒	☒		☒	☒	☒				☒		☒		☒	☒	☒		☒	☒	☒	☒					▨	▨
	Audio encoder	DV17											☒			☒					☒							☒					▨	▨
																																	▨	▨
																																	▨	▨

Fig 8. Design-Value Matrix with class 1 attributes mapped to design variables



	Deriving combinatorial value		Provide custom ringtones	Identify products	Provide VoIP telephony
			X"1	X"2	X"3
Class 0	Silenceable	X1			
	Place calls	X2			╱
	Receive calls	X3	╱		╱
	Track calls	X4	╱		╱
	Be stylish	X5			
	Be simple to use	X6			
	Be durable	X7			
	Conops style	X8			
Class 1	Tx/Rx Data	X'1	╱	╱	╱
	Provide quality audio	X'2			
	Play music	X'3	╱		
	Manage contacts	X'4			╱
	Provide calendar	X'5			
	Notify user	X'6	╱		╱
	Take pictures	X'7		╱	
	Provide light	X'8			
	Comm w/ peripherals	X'9			
	Store data	X'10	╱		
	Be low power	X'11			
	Be lightweight	X'12			
	Be small	X'13			
	Be easy to read	X'14			
	Network portability	X'15			
	Provide email service	X'16			
	Provide text message service	X'17		╱	
	Provide web service	X'18			╱
Class 2	Provide custom ringtones	X"1	■		
	Identify products	X"2		■	
	Provide VoIP telephony	X"3			■

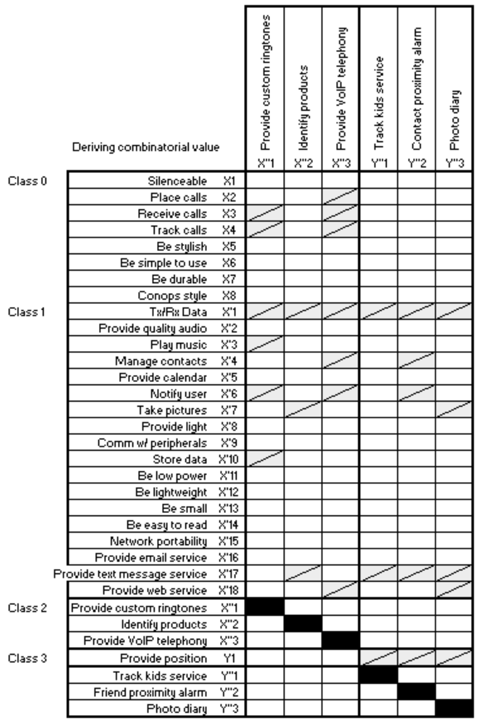
Fig 9. Class 2 Attributes derived from combining 0 and 1

Once the class 0 and class 1 attributes have been mapped, the next step is to use these attributes to derive higher order combinatorial value. Combinatorial value takes advantage of design form and function already in the system and with a "combiner" cost, creates new opportunities for value delivery. Figure 9 illustrates this example, deriving the new attributes "provide custom ringtones," "identify products," and "provide VoIP." The diagonal marks in this matrix means the attribute is "potential" and that additional cost beyond the original design concept is required to realize these connections.

The next step is for the designer to anticipate future needs of users, through either market pull or technology push means.[3] If the display of these attributes in the system requires a change to the system (addition of new design variables), then these attributes are class 3 attributes. In this example, suppose the designer notices that navigation technology has matured to the point that it can be readily integrated into phones, thereby providing users with position information. "Provide position" becomes a class 3 attribute and "navigation payload" becomes a new design variable. Diagonal marks are put in the DVM to represent the additional cost for modifying the system to allow for this type of value. Given the new class 3 attribute and the class 2 attributes, additional new class 3 attributes can be derived in a similar fashion to the class 2 attributes: through recombining to create higher order attributes. Figure 10 below illustrates with some example higher order attributes.

The new derived class 3 attributes include "track kids service," "contact proximity alarm," and "photo diary." The "track kids service" uses the "provide position" data relayed back through the data network to the cell phone service provider. Access to this data is available through subscription service [15,16]. The "contact proximity alarm" uses the position data of users linked to phone numbers in a user's contact list. When contacts reach a certain physical distance from the user, the user's cell phone provides customized notification. The "photo diary" uses the phone's ability to take photos, to provide time and position stamps, and to upload the time-placed photo directly to a user's online photo diary.

[3] <u>Market pull</u>: features or functions requested or anticipated by users; <u>technology push</u>: features or functions offered by technology development and intended to stimulate user demand.



	Deriving combinatorial value		Provide custom ringtones X"1	Identify products X"2	Provide VoIP telephony X"3	Track kids service Y"1	Contact proximity alarm Y"2	Photo diary Y"3
Class 0	Silenceable	X1						
	Place calls	X2			◩			
	Receive calls	X3	◩		◩			
	Track calls	X4	◩		◩			
	Be stylish	X5						
	Be simple to use	X6						
	Be durable	X7						
	Conops style	X8						
Class 1	Tx/Rx Data	X'1	◩	◩	◩	◩	◩	◩
	Provide quality audio	X'2						
	Play music	X'3	◩					
	Manage contacts	X'4			◩		◩	
	Provide calendar	X'5						
	Notify user	X'6	◩		◩		◩	
	Take pictures	X'7		◩				◩
	Provide light	X'8						
	Comm w/ peripherals	X'9						
	Store data	X'10	◩					
	Be low power	X'11						
	Be lightweight	X'12						
	Be small	X'13						
	Be easy to read	X'14						
	Network portability	X'15						
	Provide email service	X'16						
	Provide text message service	X'17		◩		◩	◩	◩
	Provide web service	X'18			◩			◩
Class 2	Provide custom ringtones	X"1	■					
	Identify products	X"2		■				
	Provide VoIP telephony	X"3			■			
Class 3	Provide position	Y1				◩	◩	◩
	Track kids service	Y"1				■		
	Friend proximity alarm	Y"2					■	
	Photo diary	Y"3						■

Fig 10. Class 3 Attributes derived from combining 0, 1, 2 and 3

The cell phone case application presented here provides a simple example of how attribute classes can be used to track and allocate design variables and costs, along with latent values in order to determine how to best manage both articulated and unarticulated values of individual users. Unlike probabilistic models of utility, such as those used in discrete choice analysis [17,18], the attribute class framework along with the DVM takes advantage of additional design information inherent in the particular system design's concept and parameterization, thereby allowing for insight into latent value as well as the cost for displaying new attributes.

IX. Discussion

System "success" is determined by matching the displayed system attributes to the articulated system attributes of a particular stakeholder. Inherent in this perspective is the dynamic and subjective nature of success: what is successful to one person may not be successful to another, and likewise success can vary over time. Given the dynamic and multi-perspective nature of articulated attributes, some mechanism should exist for a system to be able to alter the perception of displayed attributes from stakeholder to stakeholder as a function of time. Four operators can be used on this basis: (1) add attribute, (2) reveal attribute, (3) remove attribute, or (4) hide attribute.

When a system does not display enough value-enhancing attributes, the system may need to have attributes added, or at least revealed. *Adding* attributes is akin to altering a system in some way to enable the form and/or function described by an attribute. Class 2 and 3 attributes are enabled through adding. An example includes the personal computer, which is shipped with a set of core components and peripherals. Users can add new peripherals to add attributes, in this case often taking on new functionality. *Revealing* attributes is akin to displaying attributes; moving an attribute from a latent value to a perceived one. Class 1 attributes are enabled through revealing. An example is the case of deployed full versions of software with limited functionality. Users can purchase keys, which when entered into the software, reveal the full functionality latent in the programming.

When a system displays value-reducing or inhibiting attributes, the system may need to have attributes removed, or at least hidden. *Removing* attributes is akin to altering a system in some way to subtract form and/or function described by an attribute. An example is a software suite on personal computers sold through retail, with vastly more software than a typical user desires. Users uninstall unneeded features in order to customize their experience. *Hiding* attributes is akin to subtracting attributes from those displayed, moving the attribute from a perceived one back to a latent one. An example includes the universal remote control with a built-in physical panel for hiding extra buttons. The large number of unused buttons adds to the experienced complexity of the remote, and reduces usefulness to the user. The panel, by blocking from sight many of the specialized buttons, effectively hides the extra functionality of the remote.

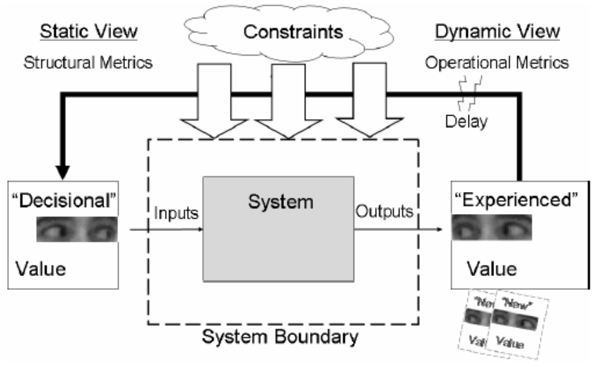


Fig. 11. Decisional versus Experienced Value.

One important aspect of change in perceived value is how the stakeholder's value perception will change with operational use of the system. As illustrated in Figure 11, these are two distinct perceptions of value. Oftentimes the requirements generation process constrains a decision maker to have only a static view of the future system. When the system comes into operation, many constraints may impact the value realization, and the experience of using the system may be very different than envisioned during concept development. Thus, the experienced value may be different from the original decisional value. Since disappointment is derived from the difference between expectations and experiences, it is important for the designer to realize and anticipate the potential value difference to the best extent possible.

The designer's challenge is to anticipate the inevitable changing needs of stakeholders. In some cases, the designer will be able to foresee future needs and provide free or combinatorial latent value as needs change. It may also be the case that the designer will need to enhance the system in order to access new value. Different stakeholders may need to access different sets of value for the same system, and the key challenge is to find resource effective ways for value perception discrimination across stakeholders. A construct that offers a mechanism for delivering combinatorial value is the *system shell,* a value robustness-creating construct for mitigating the effects of changes in context and expectations by decoupling a system from sources of change [19]. The concept of *system mask*, which is part of the system shell, may be effective for hiding or revealing attributes differentially across stakeholders.

Bearing costs of latent value is an issue that the designer will need to consider. Latent system value, which may never be desired by users, may be difficult to justify if carrying costs are not offset in the future. Strategic business decisions will need to be made to consider the net benefit of investing in attributes to deliver future value, given uncertainty in demand. Ongoing research seeks to understand system architecture choices that maximize future latent value at minimum cost.

Sometimes the presence of attributes may detract from perceived system value, such as systems with excessive features that increase the perceived cost of usage. The casual phone user seeking a basic telephony experience is turned off by the camera, music-playing, feature-rich internet phone. In cases where attributes detract from perceived value, it may be necessary to have the ability to "hide" these attributes from users. During design, costs and strategies for hiding value-detracting system attributes must be considered, along with costs to reveal these attributes if users change their needs.

The cost of designing in future value, and the cost of adding, revealing, removing, and hiding attributes, needs to be considered in trade studies and system strategies. Using the attribute classification system is anticipated to result in long run cost savings and greater system success, however, the added cost of the analysis must be considered as well. The authors are researching the effectiveness of such decision making through ongoing and planned case studies.

X. Conclusion

The system designer's challenge to create systems for delivering sustained perceived value is impacted by factors such as experience after use, changes within regulatory environment or marketplace, new technologies, and other emergent needs. Contemporary requirements elicitation processes, while sound, frequently do not adequately uncover unarticulated or latent value attributes during concept development. A method has been described that uses *attribute classes* to enhance a system designer's ability to provide for new value expectations in the context of an overall value spectrum. The formal classification of system attributes within classes of articulated value, free latent value, combinatorial latent value, accessible value and inaccessible value is a structured framework for thinking about types of value and their associated system costs. The attribute classification framework is part of a larger *Design for Value Robustness* methodology, which includes frameworks, methods, and value realization mechanisms for designing systems that can deliver sustained value in the face of changing contexts and expectations.

Acknowledgments

The authors gratefully acknowledge support for this research from the MIT Systems Engineering Advancement Research Initiative (SEAri, http://web.mit.edu/seari) and the Lean Advancement Initiative (LAI, http://lean.mit.edu).

References

[1] K.T. Ulrich and S.D. Eppinger, *Product Design and Development*. Boston, MA: Irwin McGraw-Hill, 2000.
[2] A.M. Ross, *Multi-Attribute Tradespace Exploration with Concurrent Design as a Value-Centric Framework for Space System Architecture and Design*. S.M. Technology & Policy and Aeronautics/Astronautics, Cambridge, MA: Massachusetts Institute of Technology, 2003.
[3] A.M. Ross, *Managing Unarticulated Value: Changeability in Multi-Attribute Tradespace Exploration*. Ph.D. in Engineering Systems, Cambridge, MA: Massachusetts Institute of Technology, 2006.
[4] R.L. Keeney, *Value-Focused Thinking: A Path to Creative Decisionmaking*. Cambridge, MA: Harvard University Press, 1992.
[5] K. Wiegers, *Software Requirements*. Redmond, WA: Microsoft Press, 1999.
[6] INCOSE, *Systems Engineering Handbook*. INCOSE-TP-2003-002-03.1, Seattle, WA: INCOSE, August 2007.
[7] M. Christel and K. Kang, "Issues in requirements elicitation," CMU/SEI-92-TR-12, Pittsburgh, PA: Software Engineering Institute/Carnegie Mellon, September 1992.
[8] J. Mandelbaum and D.L. Reed, *Value Engineering Handbook*, IDA Paper P-4114, Alexandria, VA: Institute for Defense Analysis, September 2006.
[9] D. von Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research*. New York, NY: Cambridge University Press, 1986.
[10] A. Sage and J. Armstrong, *Introduction to Systems Engineering*. New York, NY: Wiley & Sons, 2000.
[11] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives—Preferences and Value Tradeoffs*. Cambridge, UK: Cambridge University Press, 1993.
[12] R. de Neufville, *Applied Systems Analysis: Engineering Planning and Technology Management*. New York, NY: McGraw-Hill Co, 1990.
[13] M. Whelton and G. Ballard, "Dynamic states of project purpose: transitions from customer need to project requirements--implications for adaptive management," Proceedings of 11th Annual Conference of the International Group of Lean Construction, Blacksburg, VA, 2003.
[14] D. Kahneman, and A. Tversky, Eds. *Choices, Values, and Frames*. Cambridge, UK: Cambridge University Press, 2000.
[15] L. Magid, "GPS chips in cellphones track kids and help navigate, too," in International Herald Tribune, July 19, 2007. http://www.iht.com/articles/2007/07/19/technology/ptbasics19.1.php.
[16] Staff, "'Working late' won't work anymore: new services can track you-or your loved ones-by cell phone," in *BusinessWeek*, October 31, 2005. http://www.businessweek.com/magazine/content/05_44/b3957069.htm.
[17] J.S. Cramer, *Logit Models from Economics and Other Fields*. Cambridge, UK: Cambridge University Press, 2003.
[18] K.E. Train, *Discrete Choice Methods with Simulation*. Cambridge, UK: Cambridge University Press, 2003.
[19] A.M. Ross and D.H. Rhodes, "The system shell as a construct for mitigating the impact of changing contexts by creating opportunities for value robustness," Proceedings of the 1st Annual IEEE Systems Conference, Honolulu, HI, April 2007.