AIS Transactions on Human-Computer Interaction

Volume 8 | Issue 4 Article 1

12-2016

Can we Take User Responses at Face Value? Exploring Users' "Self-stated" and "Derived" Importance of Utilitarian versus Hedonic Software Features

Adarsh Kumar Kakar
Alabama State University, akakar@alasu.edu

Recommended Citation
Kakar, A. K. (2016). Can we Take User Responses at Face Value? Exploring Users' "Self-stated" and "Derived" Importance of Utilitarian versus Hedonic Software Features. *AIS Transactions on Human-Computer Interaction, 8*(4), 131-148. https://doi.org/10.17705/1thci.00082
DOI: 10.17705/1thci.00082

Transactions on Human – Computer Interaction

Research Paper

Can we Take User Responses at Face Value? Exploring Users' "Self-stated" and "Derived" Importance of Utilitarian versus Hedonic Software Features

Adarsh Kumar Kakar
Computer Information Systems and Alabama State University
akakar@alasu.edu

Abstract:

Empirical studies in the product development literature have shown that the users' self-reported importance of product attributes differs from the derived importance of product attributes obtained through the attributes' correlation with an external criterion such as user satisfaction. However, no study has examined this phenomenon in the context of software products. This investigation is important because the present-day software requirement-prioritization techniques are based on capturing users' self-reported importance of new software product features. As such, I develop a method in the study to capture the derived user importance of new features. The findings show that the implicitly derived importance of software attributes differs from the importance rankings assigned to them using requirement prioritization techniques. Further, I found that the implicitly derived user importance to identify the determinants of user satisfaction more accurately than the prioritization techniques based on self-stated user importance. I discuss the implications of this promising new approach for practice and future research in requirements prioritization.

Keywords: User Self-report, Implicit Method, Hedonic Features, Utilitarian Features.

1 Introduction

Gaining satisfied users begins with effectively capturing, analyzing, and understanding their genuine requirements. However, we face the question about whether we take user responses at face value. Vavra (1997) has observed that the relative importance of product attributes differs depending on whether they are explicitly stated (defined as users' self-reported importance) or implicitly derived (defined as statistically obtained by correlating attributes with an external criterion such as user satisfaction). Researchers have empirically tested this observation in the non-software product domain.

However, there is a gap in the literature: studies have examined the explicit-implicit phenomenon only for existing product attributes. No study has examined this phenomenon's applicability to new product attributes or new features that users requested, and no study has proposed theory to explain the phenomenon. Also, so far, studies have compared the implicit user importance with only one explicit method, the constant sum scale. For greater validity, we need to compare the implicit user importance with results obtained from other explicit methods (Matzler & Sauerwein, 2002).

Further, all the aforementioned investigated investigations have focused on non-software products. An investigation of this phenomenon is particularly important in the software context because it will help assess the efficacy of existing prioritization techniques in capturing software requirements that reflect "true" user needs. A review of the literature shows that, currently, all prioritization techniques capture only the users' self-stated importance. Yet, no study has investigated the "explicit-implicit" phenomenon in the context of software products for either new or for existing attributes.

Keeping this context in view, I investigate the phenomenon of users' explicit-implicit importance through an experimental study with actual users of a software product. I pose two research questions:

> **RQ1:** Are users' self-reported importance of new features and implicitly derived importance of new product features similar?
>
> **RQ2:** If not, how do we determine the "true" user importance of new software product features?

To answer these questions, I begin by developing a theory of the phenomenon based on concepts gleaned from a multi-disciplinary review of the literature. The theory builds on the widely accepted premise across disciplines that any new product feature provides value to the user along two dimensions: utilitarian and hedonic. I use the theory to hypothesize the expected outcomes. I then conduct an experiment with actual users of a software product. For this experiment, I developed a method to determine users' derived importance of new software product features. In the experiment, I compared the results of feature choices obtained from capturing users' explicit preferences using three widely used requirement engineering techniques with the results obtained from capturing users' derived preferences and the impact of the choices on user satisfaction. I discuss the findings and their implications for practice and future research.

2 Theory Development

The conceptualization of hedonic and utilitarian product values (HV and UV) as distinct and independent constructs first emerged in the consumer research literature (Holbrook & Hirschman, 1982; also see Diefenbach, Kolb, & Hassenzahl, 2014). Scholars later introduced the equivalent constructs of hedonic and pragmatic quality into the human-computer interaction (HCI) literature (Hassenzhal, Platz, Burmester, & Lehner, 2000) and intrinsic (perceived usefulness) and extrinsic motivators (perceived enjoyment) into the technology acceptance model (TAM) literature (Davis, 1989; Venkatesh, 1999).

Scholars generally agree that a product's utilitarian and hedonic dimensions are distinct and together capture its essential facets (Batra & Ahtola, 1990; Block, 1995; Dhar & Wertenbroch, 1999; Mano & Oliver, 1993; Schmitt & Simonson, 1997; Strahilevitz & Myers, 1998; Veryzer, 1995). While the product attributes that provide UV are functional and goal oriented and generate cognitive response from the user, the product attributes that provide HV represent novelty, aesthetics, unexpectedness, pleasure and fun and evoke affective user responses (Strahilevitz & Myers, 1998).

HV derived from using a product provides intrinsic motivation to users. Intrinsically motivated users experience pleasure and enjoyment regardless of the external benefits that the UV provides them such as improvement in productivity, compensation, and career progression (Cocosila, Archer, & Yuan, 2007; Venkatesh & Speier, 1999; Deng, Doll, & Troung, 2004; Li & Hsieh, 2007; Saade, 2007). However, HV is still important for users of utilitarian products and organizations that deploy them. Research has found that

intrinsic motivators lead to users' greater commitment to using a system (Li & Hsieh, 2007; Venkatesh, Speier, & Morris, 2002; Saade, 2007), which may lead to better performance (Li & Hsieh, 2007). Intrinsically motivated users experience a sense of flow, which refers to feelings of playfulness, enjoyment, and cognitive absorption (Ha, Yoon, & Choi, 2007; Hsu & Lu, 2004; Liu, Liao, & Pratt, 2009; Gerow, Ayyagari, Thatcher, & Roth, 2013). While users consider the extrinsically motivating UV a means to an end, they consider the intrinsically rewarding HV as an end in itself (Ha et al., 2007).

However, past research shows that, when users have to choose between a hedonic and a utilitarian alternative, they show their preference for the utilitarian (Diefenbach & Hassenzahl, 2011). Research has termed the reason underlying this phenomenon as "lay rationalism" (Lindgaard, Fernandes, Dudek, & Brown, 2006). People wish to make a rational choice. However, hedonic features are hard to justify rationally compared with utilitarian features (Diefenbach & Hassenzahl, 2009). Research has associated further hedonic attributes with waste, luxury, and guilt (Kivetz & Simonson, 2002; Hsee, Zhang, & Xi, 2003). Users wish to be seen as addressing points of pain before addressing their pleasure needs to avoid guilt (Berry, 1994) even though they may prefer the latter (Diefenbach & Hassenzahl, 2011). Thus, the "explicit" importance of user requirements obtained through self-reports may reflect "socially acceptable" or "politically correct'' responses (Oliver, 1997). As such, they will not likely match the "derived" response when one mitigates the effects of the aforementioned causes that distort users from expressing their "true" choice. As such, I hypothesize that:

Hypothesis 1: The explicit importance that users' assign to a new product feature differs from its implicit importance to the user.

Traditionally, the techniques used for software feature selection have relied on the explicit methods such as users' (and sometimes other stakeholders such as development team members') self-reports. Table 1 summarizes and briefly describes the commonly cited feature selection methods from the requirement engineering literature (adapted from Berander & Andrews, 2006). Therefore, all these methods are subject to the aforementioned biases associated with explicit methods. To resolve this predicament, one can work backwards by first obtaining users' overall satisfaction with the proposed implementation of a subset of requirements that the users choose and then using an appropriate statistical technique to derive the implicit importance of the individual requirements to the users. To accomplish this goal, I adapt the widely accepted (Brandt, 1987; Anderson & Mittal, 2000; Brandt, 1988; Mittal, Ross, & Baldasare, 1998) penalty reward contrast analysis (PRCA) technique in the context of new feature selection

Table 1. Methods of Feature Selection from Requirements Engineering Literature (Adapted from Berander & Andrews, 2006)

Technique	Description
Priority groups (Wiegers, 1999)	In this grouping technique, users group software requirements into user priority categories, such as low (nice to have), medium (regular), and high (critical).
Planning game (Beck, 2001)	In this technique, the development team (which includes a user representative) sorts the requirements by effort, risk, and value. Based on the relative assessments, the scope of the next release is set.
100 points (Leffingwell & Widrig, 2003)	In this ranking technique, each stakeholder allocates gives each requirement up to a total of 100 points. One then ranks the requirements by sorting them based on the total points allocated by all the stakeholders.
Theory W (Boehm & Ross, 1989)	In this technique, each stakeholder notes which requirements they would be willing to remove and which are most important. Through negotiation among themselves, the stakeholders then determine the prioritized list.
AHP (Saaty, 1980)	This technique addresses situations involving multi-criteria decision making by comprehensively comparing the cost and value of each requirement pair to determine the priority list.
B-tree prioritize (Heger, 2004)	This technique uses an algorithm to economize on the number of pair wise comparison of requirements in a given set of candidate requirements for determining a prioritized list.
Value-oriented prioritization (Azar, Smith, & Cordes, 2007)	In this method, the company executives first identify the core business values and then rank these values on a relative scale. The executives than generate a ranked list of requirements by assigning each requirement a weight in each value category.

By having two sets of dummy variables representing each feature request with a value of (1, 0) indicating the user preference for its implementation and a value of (0, 1) indicating the user choice for its non-implementation, multiple regression analysis is conducted with the overall user satisfaction with the feature subset as the dependent variable. Using this statistical approach, we get two regression coefficients for each feature, one representing the user satisfaction with implementation of a feature and the other regression coefficient representing the user (dis)satisfaction with non-implementation of the same feature. The sum of the absolute values of the significant regression coefficients for implementation and non-implementation of a feature represents the assessment of users' implicit importance of a feature by providing a measure of the range of impacts of a feature on overall user satisfaction (Mikulic & Prebez, 2008).

Capturing users' overall satisfaction for a feature set and then deriving the importance of individual features in terms of their impact on user satisfaction helps obfuscate the user need for providing a socially acceptable response. Further, this method mitigates the guilt feelings of indulging in luxury (or waste) associated with users' self-reporting their explicit preference of specific individual features. Thus, the statistically derived implicit importance of individual features is more likely to accurately reflect user' true needs. As such, I hypothesize that:

Hypothesis 2: Users' satisfaction with implementing an implicitly derived feature set is higher than users' satisfaction with implementing the feature set selected using explicit methods.

3 Method

I used an experimental method in the study. Experimental studies are useful in systematically identifying the differential impacts of treatments (features selected using various feature selection techniques) on outcome variables (user satisfaction).

Table 2. Sample User Feature Requests Used in the Study

	Feature description
1	**Choose from a calendar** Allow dates to be chosen from a calendar
2	**Shortcut to create tasks** Enable users to create a shortcut to go directly into the task creation mode.
3	**Make Quiet Hours completely quiet** Have a new option—"super quiet hours"—during which all reminders should be disabled
4	**Purging completed tasks** Provide a feature to purge all completed tasks.
5	**Color tasks based on priority** Enable users to visually see task priority through a color coding scheme
6	**Due Date only** The application should allow the user to bypass the due time
7	**Creating tasks that repeat yearly** Allow creation of yearly recurring tasks to remind users about important events such as birthdays, and anniversaries
8	**Grocery shopping list** Provide a feature to enable users to create and update a regular grocery list
9	**New task default priority** All tasks should be set to medium priority by default.
10	**Geolocation reminders** Provide a feature to remind users of that they are passing through an important geolocation

3.1 Experimental Setting

I chose Astrid Task Manager, a popular mobile application, as the software product to investigate for this study. At the time I conducted the study, Astrid had an active user forum where users could post and provide their comments on new feature requests. Actual users of Astrid Task Manager provided their

responses to a pen and paper-based survey in January, 2013. I used 10 randomly selected feature requests (see Table 2) made by users of Astrid Task Manager as the test instrument.

3.1.1 Subjects

I used a representative user group of young adults (ages 19-24) as subjects. Researchers recognize users in this age group as early adopters of the latest technologies and responsive to innovations (new features in my experimental context) (Ehrenberg, Juckes, White, & Walsh, 2008). One hundred and forty-eight subjects participated in the experiment from which I obtained 128 valid responses. The valid responses from 72 males (average age 21.6 years) outnumbered the valid responses from 56 females (average age 21.7 years). All subjects actually used Astrid, and I recruited them from a state university. I took their consent for participating in the study.

I trained the subjects on the methods of feature selection used in the experiment just before I conducted the study. I explained the 10 Astrid feature requests in the experiment by reading out each feature request aloud and seeking subject response on whether they required any clarification. The effect size found during the pilot study in October, 2012, with 38 users of Producteev, another task-planning software, helped determine the sample size. Looking up Cohen's power primer (Cohen, 1992) for a medium-sized effect observed during the pilot and assuming a power of 0.8 and alpha = 0.05 (one tail) gave a sample size of 54 subjects.

3.2 Selection of Methods

I selected three methods of prioritizing requirements from the software requirements engineering literature for the experimental investigations in the study. In Sections 3.2.1 to 3.2.3, I describe the reasons for selecting these methods from among the methods listed in Table 1.

3.2.1 Binary Search Tree

Racheva, Daneva, and Buglione (2008), who classified requirements-prioritization techniques according to their scalability, found that the Binary search tree method scales up well up to medium-scale requirements. Bebensee, van de Weerd, and Brinkkemper (2010) have observed that, for market-driven software products, one can expect a larger number of new feature requests from users than software products developed for a single customer. Hence, techniques that work well for only small-scale requirements may not be suitable for software products for mass markets. In their comparative study, Ahl (2005) found that the Binary search tree performed better on many criteria including scalability and accuracy of results compared to the other four techniques of requirements prioritization (i.e., planning game, AHP, planning game combined with AHP (PGCAHP)) and the 100 points method).

3.2.2 Priority Groups Method

Industry professionals working in a market-driven product development environment prefer simple tools (Lehtola & Kauppinen, 2006; Berander & Andrews, 2006). However, many large-scale or even medium-scale requirements prioritization techniques are based on relatively complex algorithms (Rachdeva, Daneva, & Buglione, 2008). This complexity may account for the popularity of the simple-to-use priority groups method (Wiegers, 1999). It is among the most traditional and best-known methods of requirements prioritization recommended by the IEEE (Sillitti & Succi, 2006) that classifies requirements into three priority groups: high, medium and low (Lehtola & Kauppinen, 2006).

3.2.3 Constant Sum Scale or 100 points Method

In their empirical study, Griffin and Hauser (1993, p. 17) considered three different measures of importance (direct rating, constant-sum scale, anchored scale) and found no significant differences between the methods. However, constant sum scales more sharply distinguishes importance weights. Because explicit-implicit studies have used this scale (e.g., Matzler and Sauerwein, 2002), I also measured explicitly derived importance using a constant-sum scale: the 100 points method.

3.3 Experimental Treatments

I describe the process that the subjects used in arriving at a prioritized feature subset from the set of 10 Astrid user feature requests using each of the different prioritization techniques below.

Binary search tree method: subjects created a ranked list of user requirements by first creating a node with the first requirement in the requirement set and comparing the next requirement with this node. If the requirement is of lower priority it is placed on the left of the node else it is placed on the right of the node. This process continues with subsequent requirements in the requirement set until a ranked list of requirements is produced. The node at the extreme right is the requirement of highest priority to that subject and the node at the extreme left is the requirement of lowest priority.

Priority groups method: the following definitions (Wiegers, 1999) were used by the subjects for categorizing requirements in the High, Medium and Low categories:

High priority requirements are mission critical requirements; they are required for the next release.

Medium priority requirements support necessary system operations; they are required eventually but could wait until a later release.

Low priority requirements are a function or quality enhancement; they would be nice to have someday if resources permit.

The relative priority ranking of requirements was determined by High > Medium > Low and within each category by the number of users who opted for the requirement to belong to that category.

Constant sum scale or 100 points method: users were given 100 points to be distributed among the new feature requests based on their importance to the users. The higher the points assigned the greater is the importance of the feature to the user. All points assigned to the features should total to exactly 100. The percentage of the points assigned to the feature gives the measure of its importance. The relative ranking of the features is done by sorting them in the descending order of the percentage points assigned to them.

Implicit method: users rated their overall satisfaction with the requirements prioritized by users using the above techniques. The importance of the requirements was determined based on the regression coefficients obtained by conducting the multiple regression analyses described in detail in the theory development section.

3.4 Experimental Procedure and Measures Used

I conducted the experiment in three rounds with a weekly gap between them. Temporal separation in data collection helps mitigate potential effects due to common method variance because the same subjects provided data on independent and dependent variables (Sharma, Yetton, & Crawford, 2009). Further, it provided the time required to analyze data for tailoring the questionnaire specifically to individual subjects in subsequent rounds.

3.4.1 Round1

In round 1, 148 subjects prioritized Astrid's 10 feature requests by providing their responses to the questions, definitions, and procedures related to related to the binary search tree method, priority grouping method, and the 100 points method. The subjects categorized the requirements into two sets of requirements: one for implementation into the product and the other for non-implementation. The priority groups method provides both a subset of requirements that are important for users and which they want implemented in the software (high and medium categories) and a subset of requirements that they do not care about (low category). The other methods provide a list of requirements ranked in the order of their importance to the users but with no guidance on where the cut-off point lies for separating the features into the implementation set and the non-implementation set. As such, I used the priority groups method as a baseline. Using the priority groups method first, I identify "n" requirements (high + medium) that are important to the users. Based on this known value of "n", I then identify the top ranked "n" requirements using the tree method and 100 points method for the purpose of classifying them in the implementation subset. Finally, I include the bottom "10-n" requirements using these three methods in the non-implementation subset.

3.4.2 Round 2

Subjects in round 2 provided their satisfaction responses to the implementation and non-implementation feature subsets that I determined by analyzing subjects' individual responses using the three explicit techniques in round 1. I define user satisfaction as an affective user response to the degree to which the

feature subset under consideration meets the user's specified requirements. Subjects used Andrews and Withey's (1976) seven-point single-item scale (1: terrible, 2: unhappy, 3: mostly dissatisfied, 4: neither satisfied nor dissatisfied, 5: mostly satisfied, 6: pleased, 7: delighted) to rate their overall satisfaction level with the implementation subset.

3.4.3 Round 3

I determined the implementation and the non-implementation subsets using the implicit method described in the theory development section for each subject after round 2 based on the responses obtained from the subjects in round 1 and round 2. In round 3, each subject provided their user satisfaction response for the implementation and non-implementation subsets using the implicit method. Additionally, in this round, users rated the features on the hedonic-utilitarian measure (Leclere, Schmitt, & Laurette, 1994). The ratings were anchored at 1 = predominantly hedonic and 9 = predominantly utilitarian. I defined the term hedonic for the users as "something that is enjoyable and appeals to the senses" and the term utilitarian as "something that is useful, practical, functional, something that helps achieve a goal" (Dhar & Wertenbroch, 1999).

I used single-item measures for both hedonic-utilitarian and user satisfaction responses from the user in rounds 3 and 2, respectively. Single item measures are short, easy to administer, less time consuming, and not monotonous to complete (Gardner, Cummings, Dunham, & Pierce, 1998; Pomeroy, Clark, & Philip, 2001). They reduce response biases and are, therefore, relevant for use in large-scale studies (Robins, Hendin, & Trzesniewski, 2001).

3.4.4 Control Procedures

To mitigate the effect of extraneous variable on the study's findings such as segmental differences among users, I used a homogeneous sample of student subjects for the study. I controlled the sequence effect of experimental treatments by a counterbalancing study design using Latin squares (Sheehe & Bross, 1961). Every fourth subject in rounds 1 and 3 got the same sequence (see Table 5) for providing their responses in rounds 1 and 2. I randomly chose a representative set of 10 feature requests in the survey instrument by randomly selecting them from among pending feature requests of Astrid's users. To avoid variable and confounding subject responses due to shifts in structure, content, and format, I re-worded all the 10 feature requests in a simple and standard style (see Table 2). I controlled the "individual differences" among subjects in the sample by using the repeated measure design. I repeated the measurement of dependent variable (user satisfaction) in round 2 with individual subjects providing the "user satisfaction" responses to their own implementation and non-implantation feature sets determined by using the three methods of prioritization (i.e. priority group, binary search tree and 100 point method) in round 1.

Table 3. Sequencing of Methods in Round 1

Subject 1	Priority groups	Binary Tree	100 Points
Subject 2	Binary Tree	100 Points	Priority group
Subject 3	100 Points	Priority group	Binary Tree

4 Results and Analysis

Analyzing the data I obtained in round 1 (Table 4) showed that, on the aggregate, users selected six features for implementation using the priority groups method. For comparison, Table 4 shows the top six high-priority requirements that the users identified using the other two explicit methods. I identified the top six requirements that users identified with the implicit method by regressing the users' overall satisfaction with the feature subset that each method identified (see Table 5). I obtained two regression coefficients (I describe the method provided in Section 3.4.1) for each user requirement, one for implementation and another for non-implementation. I then added the absolute value of the statistically significant regression coefficients for implementation and non-implementation of a feature to determine the six features in the subset for implementation.

Table 4. Results of Astrid Features Selected Based on Data Collected in Round 1

	Feature number	1	2	3	4	5	6	7	8	9	10
100 points method	Ranks 1-6	**79**	**73**	33	**94**	**85**	**70**	55	41	**77**	25
	Ranks 7-10	49	55	95	34	43	58	73	87	51	103
Priority groups method	High	**58**	50	**83**	34	31	46	20	22	**62**	23
	Medium	33	10	9	**63**	**78**	**56**	16	17	20	33
	Low	37	70	36	31	19	26	73	89	48	72
Binary search tree method	Ranks 1-6	**92**	**86**	**75**	**67**	**81**	**87**	40	47	35	59
	Ranks 7-10	36	42	53	61	47	41	88	81	93	69

Consistently, across the three explicit methods, the features selected for implementation and non-implementation in the software product using this implicit method were the same (Table 5). The sum of the absolute value of regression coefficients for the six features numbered as 1, 2, 4, 7, 9, and 10 were higher across the three methods and, therefore, qualified for implementation in software. The sum of the absolute value of regression coefficients for the four features numbered as 3, 5, 6, and 8 were lower (in fact, 0 as none of the regression coefficients were significantly different than 0) across the three methods and, therefore, did not qualify for implementation in software.

Table 5. Results of Astrid Features Selected after Multiple Regression Analysis

Feature number		1	2	3	4	5	6	7	8	9	10
Binary search tree method	I	**0.016**	**.980***	0.003	**0.009**	0.014	0.008	**1.010***	0.013	**1.450***	**1.050***
	NI	**-.390***	**-.580***	-0.004	**-.590***	-0.009	0.013	**0.027**	0.005	**-1.270***	**0.019**
100 points method	I	**0.011**	**.790***	0.018	**0.017**	0.034	0.028	**1.020***	0.033	**1.290***	**1.680***
	NI	**-.480***	**-.870***	0.075	**-.920***	-0.021	0.019	**0.014**	0.007	**-.1.160***	**0.036**
Priority groups method	I	**0.022**	**.850***	0.029	**0.046**	0.072	0.036	**1.200***	0.063	**1.290***	**1.430***
	NI	**-.630***	**-.890***	-0.047	**-.940***	-0.058	0.041	**0.065**	-0.011	**-1.120***	**0.170**

* $p < 0.05$

Table 6 summarizes the feature numbers (see Table 1 for feature description) that various methods selected (indicated with an "x") for implementation.

Table 6. Astrid Features Selected Using the Four Different Methods

Feature number	1	2	3	4	5	6	7	8	9	10
100 points	x	x		x	x	x			x	
Priority group	x		x	x	x	x			x	
Binary search tree	x	x	x	x	x	x				
Implicit	x	x		x		x	x		x	x

From Table 6, we can see that the set of six features selected by explicit and implicit methods in the implementation set differ. I found only one feature selected to be different in the implementation set when comparing the result of any explicit method with any other explicit method. However, in the case of implicit method, at least two features selected in the implementation set were different compared with those selected using any of the three explicit methods. Thus, these results support Hypothesis 1.

Further, from Table 7, we can see that two features 7 and 10 were 1 standard deviation below mean on the hedonic-utilitarian scale, indicating these features were predominantly hedonic. As expected, both these features were not consistently selected for implementation by the explicit methods with the exception of priority groups method. Further, as predicted, both these features were consistently selected for implementation by , method (see Table 10). These findings support my arguments leading up to Hypothesis 1 in the theory development section that users find hedonic features hard to justify compared with the utilitarian features even though their true preference is for the former. Therefore, in general, explicit methods using self-reports will not be as efficacious in selecting hedonic features for implementation into the software as they are in selecting utilitarian features (features 1 and 4, see Tables 6 and 7). By contrast, I found the implicit method to be efficacious in selecting both hedonic and utilitarian features (Tables 6 and 7).

Table 7. Feature Wise Rating on the Hedonic-Utilitarian Scale

Feature number	1	2	3	4	5	6	7	8	9	10
Hedonic (1 SD < mean)							0.13			0.08
Hybrid (>=1 SD and <= 1 SD)		0.45	0.58		0.38	0.59		0.35	0.48	
Utilitarian (1 SD > mean)	0.81			0.78						
SD = standard deviation										

Table 8 shows the results of users' self-stated overall satisfaction responses in round 2 and 3 with the feature set obtained by the four different methods. I analyzed the results using repeated measure ANOVA (see Table 9) and found that, while there was no significant difference between the feature sets obtained from the three explicit methods, the user satisfaction with the feature set selected by the implicit method was significantly higher than the user satisfaction with the feature set obtained from all other methods, which supports H2.

Table 8. Overall User Satisfaction (Self-stated) with Implementation Feature Subsets

Experimental conditions	Mean user satisfaction	Standard deviation
Implicit (1)	5.680	0.534
Priority groups method (2)	4.913	0.683
100 points method (3)	4.951	0.582
Binary Search tree method (4)	4.956	0.665

Table 9. Difference in User Satisfaction (Self-stated) with Implementation Feature Subset

Method	1	2	3	4
Implicit (1)	0			
Priority groups method (2)	0.372*	0		
100 points method (3)	0.334*	-0.038	0	
Binary Search tree method (4)	0.329*	-0.043	-0.005	0
* $p < 0.05$				

I derived the overall user satisfaction for implementing the feature set obtained by adding the two regression coefficients of features selected for implementation using the implicit and explicit methods from Table 5 and present it in Table 10. For the implicit method, I obtained the regression coefficients for implementing the feature set (I) by taking the average of the regression coefficients (I) obtained from regressing the overall user satisfaction with the feature set using the three explicit methods (see Table 5). I analyzed the data in Table 10 using repeated measure ANOVA (see Table 11) and found that the implicitly derived overall user satisfaction with the feature set selected by the implicit method was significantly higher than the implicitly derived overall user satisfaction with the feature set obtained from all other methods. Thus, these results support H2.

Table 10. Overall User Satisfaction (Derived) with Implementation Feature Subsets

Experimental conditions	Mean user satisfaction
Implicit (1)	4.49
Priority groups method (2)	2.14
100 points method (3)	2.47
Binary search tree method (4)	0.98

Table 11. Difference in User Satisfaction (Derived) with Implementation Feature Subset

Method	1	2	3	4
Implicit (1)	0			
Priority groups method (2)	2.350***	0		
100 points method (3)	2.020***	0.330	0	
Binary search tree method (4)	3.510***	-1.160***	-1.490***	0
*** $p < 0.001$				

5 Discussion

This study's findings fully support both H1 and H2. The implicit method of feature selection better elicits users' "true" needs than the explicit methods. Explicit methods using users' self-reports have limitations. Users are hesitant to choose hedonic features at a rational level although they may prefer to acquire them. In contrast, utilitarian features are easier to justify than hedonic features. Users like others to see them as addressing their practical functional needs rather than indulging in pleasure. This behavior became clear when the explicit methods using user self-reports selected both the predominantly utilitarian features with greater efficacy than they did in selecting the two predominantly hedonic software features (see Tables 6 and 7). Yet, hedonic features (7, 10) had a higher positive impact on user satisfaction than utilitarian features (1, 4) (see Table 5).

In contrast, the implicit method accurately selected both hedonic and utilitarian features for implementation (see Tables 6 and 7). The implicit method's ability to find users' genuine needs is not surprising considering that implicit methods do not embarrass users or create guilt by exposing their preference for hedonic features. Further, the implicit method does not overlook their needs for utilitarian features. While hedonic features represent Herzberg, Mausner, and Snyderman's (1967) motivators, utilitarian features represent Herzberg et al.'s (1967) hygiene factors. Hedonic features are satisfiers (i.e., implementing them leads to satisfaction but not implementing them does not cause dissatisfaction) and utilitarian features are dissatisfiers (i.e., implementing them does not cause user satisfaction but not implementing them causes dissatisfaction) (Zhang & von Dran, 2002). By identifying two regression coefficients representing user satisfaction with implementing a feature and user dissatisfaction with not implementing a feature and summing up their absolute values for determining the relative importance of features enables the implicit method to identify salient hedonic (7, 10) and utilitarian features (1,4) (see Table 5). Further, this approach also enables the implicit method to identify salient hybrid features (2, 9) (Table 5).

Hybrid features are neither satisfiers nor dissatisfiers but, due to their hybrid nature, cause user satisfaction on implementation and user dissatisfaction on non-implementation. These feature characteristics are in line with the theory of attractive quality (Kano, Seraku, Takahashi, & Tsuji, 1984)—an extension of Hezberg's (Herzberg et al., 1967) motivation-hygiene perspective—that identifies three types of features: must-be, one-dimensional and attractive instead of just satisfiers (motivators) and dissatisfiers (hygiene). From the perspective of the theory of attractive quality, while must-be features are dissatisfiers, attractive features are satisfiers and one-dimensional features are both.

Must-be features are prerequisite features: users assume the product will provide them. Thus, must-be features lead to user dissatisfaction on non-implementation but no satisfaction on implementation. Intuitively, it makes sense for users in the study to classify "choosing date from a calendar" in the must-be/utilitarian/dissatisfier category since the feature is so basic. Further, the feature is commonly available across applications, and users do not expect to enter dates manually. But, at the time of the study, Astrid

did not provide this core feature. Thus, although users would likely be dissatisfied if software does not meet this must-be requirement, they do not think it is a big deal.

Attractive features are those that surprise users pleasantly and create delight. Implementing attractive features in a product enhances user satisfaction but causes no dissatisfaction on non-implementation. At the time of the study, the "Geolocation reminders" feature would have excited the users enough for them to classify it in the predominantly hedonic (attractive/ satisfier) category. Yet, while the unexpected value that implementing this feature in the software product provides enhances user satisfaction, it is not likely to cause user dissatisfaction on non-implementation. Implementing one-dimensional features causes satisfaction on implementation and dissatisfaction on non-implementation. Classifying "color tasks based on priority" in the hybrid/one-dimensional category has face value. Users would find this feature to be helpful in using Astrid by visually (appeal to the senses) providing them with help to identify and manage priority tasks (utility).

Thus, from the theory of attractive quality's perspective, one can consider predominantly hedonic features as attractive features, predominantly utilitarian features as must-be features, and hybrid features as one-dimensional features. The results of the study empirically confirm this association (see Tables 5 and 7). The hedonic features (7, 10) identified by the implicit method impacted user satisfaction positively on implementation but did not impact user satisfaction on non-implementation. The utilitarian features (1, 4) impacted user satisfaction negatively on non-implementation but had no impact on user satisfaction on implementation. The hybrid features (2, 9) impacted user satisfaction negatively on non-implementation and impacted user satisfaction positively on implementation. Thus, the findings (summarized in Tables 12 and 13) agree with those that one can expect from the two factor theory (Herzberg et al., 1967) and the three factor theory (Kano et al., 1984).

Further, the implicit method could also accurately identify features 3, 5, 6, 8 that are not salient from users' perspective. These features neither impacted user satisfaction significantly on implementation nor caused user dissatisfaction on non-implementation. Thus, this study's findings empirically support the theory and the approach for implicitly selecting software features proposed in the study and its arguments that the implicit method can be useful in identifying software users' "true" needs.

Table 12. Impact of Feature Implementation on User Satisfaction

Feature Types	Impacts on user satisfaction		
	Positive	No impact	Negative
Hedonic features (motivators / satisfiers / excitement)	√		
Hybrid features (performance)	√		
Utilitarian features (hygiene / dissatisfiers / excitement)		√	
Indifferent		√	

Table 13. Impact of Feature Non-Implementation on User Satisfaction

Feature types	Impacts on user satisfaction		
	Positive	No impact	Negative
Hedonic features (motivators / satisfiers / excitement)		√	
Hybrid features (performance)			√
Utilitarian features (hygiene / dissatisfiers / excitement)			√
Indifferent		√	

6 Contribution

Research have long suggested involving users as active contributors in the product-development process rather than as passive participants (Gardiner & Rothwell, 1985; Leonard-Barton, 1995; Rothwell, 1976; von Hippel, 1988; Witell, Lofgren, & Gustaffsson, 2011). Collectively, users constitute a source of much

product innovation (Vonn Hippel, Ogawa, & de Jong, 2012). When one views users merely as recipients of innovation, firms do not have access to user knowledge and experience developed through product use (Sawhney, Verona, & Prandelli, 2005).

Therefore, to actively engage users, organizations have evolved various mechanisms. Of these mechanisms, many organizations have increasingly begun to use websites for capturing and prioritizing user requirements. The websites include both forums and collaborative tools and allow large numbers of users to participate in the requirements-gathering and analysis process. The success of websites and collaborative tools that have been used to gather inputs from users demonstrates that, given the opportunity, users too are willing to take the time to contribute feedback and ideas (Laurent & Cleland-Huang, 2009).

However, whatever the mechanism used, by actively engaging users, one often elicits more new feature requests than one needs to build into the system. The large number of feature requests creates a dilemma for software-development organizations. While, on the one hand, excluding a high-value feature may mean losing users to a competing product, on the other hand, including an unneeded requirement creates wasted development effort, delays the time the product takes to reach the market, and increases the product's complexity, maintenance, and operational costs. Research has suggested that a product's evolution should be innovative in the users' frame of mind, not the developers' (Fellows & Hooks, 1998). Implementing new product features that do not resonate with the users may result in product investments that are counterproductive.

Perhaps for this reason, requirements prioritization methods have traditionally used user self-reports to identify features to implement into software. However, this study's findings questions whether we can take user responses to new product features at face value. Previous studies have suggested that self-reported user preferences are likely to be biased. However, to the best of my knowledge, no study has investigated this phenomenon for software products. Further, no study in the software or non-software product domains has investigated this phenomenon theoretically and empirically validated the proposed theory. In this study, I found evidence to suggest that we cannot take the user self-report at face value. While users' self-reports can efficiently identify utilitarian features, they do have a problem with correctly identifying hedonic features, which highlights a critical issue in software-feature selection.

In a recent meta-analysis of technology adoption model (TAM) studies, Gerow et al. (2013) note that both utilitarian and hedonic features are equally important to the users of even utilitarian products. As such, software-development organizations need to accurately identify salient utilitarian and hedonic features to implement in their product. By adapting the well-known penalty contrast reward analysis (PRCA) method to select new features, this study provides an implicit method for capturing hedonic and utilitarian features that users find important. I tested this method with actual users of a software product, and it outperformed promising existing methods from requirements engineering domain that prioritize requirements based on users' self-reports.

The method developed in this study could not only identify salient predominantly hedonic and predominantly utilitarian features but also salient hybrid features that can impact user satisfaction. Further, the study also accurately identified features that users do not care about. These findings are useful for practitioners because they will help in managing user satisfaction and maximizing return by suggesting which features that users value to implement and eliminating those they do not. Further, the findings provide practitioners with information on prioritizing features under conditions of resource constraint. Depending on product goals and the resources available, software product managers can make decisions to include or exclude a new feature for implementation into the product.

For example, if one seeks to first preclude dissatisfaction, then one should prioritize implementing requirements 9, 4, 2, and 1 in that order of priority depending on the availability of resources. If one seeks to enhance user satisfaction, then one should focus on implementing features 9, 10, 7, 2 in that order of priority. Further, if one seeks to enhance a product's hedonic value by making it more attractive and enjoyable to use, then one should focus on implementing features 7 and 10. If one seeks to enhance a product's utilitarian value by improving user productivity and helping users meet their functional goals, then one should focus on implementing features 1 and 4.

References

Ahl, V. (2005). *An experimental comparison of five prioritization methods* (master's thesis). School of Engineering, Blekinge Institute of Technology, Suecia.

Anderson, E. W., & Mittal, V. (2000). Strengthening the satisfaction-profit chain. *Journal of Service Research*, *3*(2), 107-120.

Andrews, F. M., & Withey, S. B. (1976). *Social indicators of well-being.* New York: Plenum Press.

Azar, J., Smith, R., & Cordes, R. (2007). Value-oriented requirements prioritization in a small development organization. *IEEE Software*, *24*(1), 32-37.

Beck, K. (2001). *Extreme programming: Explained*. Boston, MA: Addison-Wesley.

Bebensee, T., van de Weerd, I., & Brinkkemper, S. (2010). Binary priority list for prioritizing software requirements. In *International working conference on requirements engineering: foundation for software quality* (pp. 67-78). Springer: Berlin.

Berander, P., & Andrews, A. (2006). Requirements prioritization. In A. Aurum & C. Wohlin (Eds.,), *Engineering and managing software requirements* (pp. 69-94). Berlin: Springer.

Batra, R., & Ahtola, O. T. (1990). Measuring the hedonic and utilitarian sources of consumer attitudes. *Marketing Letters, 2*(2), 159-170.

Berry, C. J. (1994). *The idea of luxury.* Cambridge: UK: Cambridge University Press.

Block, P. (1995). Seeking the ideal form: Product design and consumer response. *Journal of Marketing, 59*, 16-29.

Boehm, B., & Ross, R. (1989). Theory W software project management: Principles and examples. *IEEE Transactions on Software Engineering*, *15*(7), 902-916.

Brandt, D. R. (1987). A procedure for identifying value-enhancing service components using customer satisfaction survey data. In C. Suprenant (Ed.), *Add value to your service* (pp. 61-65). Chicago: American Marketing Association.

Brandt, D. R. (1988). How service marketers can identify value-enhancing service elements. *Journal of Services Marketing, 2*(3), 35-41.

Cocosila, M., Archer, N., & Yuan, Y. (2007) Adoption of SMS for business to-consumer usage: Supporting adherence to healthy activities. In *Proceedings of the Americas Conference on Information Systems* (pp. 1-10).

Cohen, J. (1992). A power primer. *Psychological Bulletin, 112*, 155-159.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly, 13*(3), 319-340.

Deng, X., Doll. W. J., & Troung, D. (2004). Computer self-efficacy in an ongoing use context. *Behaviour & Information Technology, 23*(6), 395-412.

Dhar, R., & Wertenbroch, K. (1999). Consumer choice between hedonic and utilitarian goods. *Journal of Marketing Research, 37*(1), 60-71.

Diefenbach, S., & Hassenzahl, M. (2009). The “beauty dilemma”: Beauty is valued but discounted in product choice. In *Proceedings of the CHI 2009 Conference on Human Factors in Computing Systems* (pp. 1419-1426). New York: ACM.

Diefenbach, S., & Hassenzahl, M. (2011). The dilemma of the hedonic—appreciated, but hard to justify. *Interacting with Computers, 23*(5), 461-472.

Diefenbach, S., Kolb, N., & Hassenzahl, M. (2014). The hedonic in human-computer interaction: History, contributions, and future research directions. In *Proceedings of the 2014 Conference on Designing Interactive Systems* (pp. 305-314).

Ehrenberg, A., Juckes, S., White, K. M., & Walsh, S. P. (2008). Personality and self-esteem as predictors of young people's technology use. *CyberPsychology & Behavior, 11*(6), 739-741.

Fellows, L., & Hooks, I. (1998). A case for priority classifying requirements. In *Proceedings of the* 3rd *International Conference on Requirements Engineering* (pp. 62-65).

Gardner, D. G. Cummings, L. L., Dunham, R. B., & Pierce, J. L. (1998). Single-item versus multiple-item measurement sales: An empirical comparison. *Educational and Psychological Measurement, 6*, 898-915.

Gardiner, P., & Rothwell, R. (1985). Tough customers: Good designs. *Design Studies, 6*(1), 7-17.

Gerow, J. E., Ayyagari, R., Thatcher, J. B., & Roth, P. L. (2013). Can we have fun@ work? the role of intrinsic motivation for utilitarian systems. *European Journal of Information Systems,* 22(3), 360-380.

Griffin. A., & Hauser, J. R. (1993). The voice of the customer. *Marketing Science*, *12*(1), 1-28.

Ha, I., Yoon, Y., & Choi, M. (2007). Determinants of adoption of mobile games under mobile broadband wireless access environment. *Information and Management, 44*(3), 276-286.

Hassenzhal, M., Platz, A., Burmester, M., & Lehner, K. (2000). Hedonic and ergonomic quality aspects determine a software's appeal. In *Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems* (pp. 201-208).

Heger, D. A. (2004). A disquisition on the performance behavior of binary search tree data structures. *European Journal for the Informatics Professional, 5*(5).

Herzberg, F., Mausner, B., & Snyderman, B. B. (1967). *The motivation to work* (2nd ed.). New York: Wiley.

Holbrook, M. B., & Hirschman, E. C. (1982). The experiential aspects of consumption: Consumer fantasies, feelings, and fun. Journal *of Consumer Research, 9*(2), 132-140.

Hsee, C., Zhang, J., Yu, F., & Xi, Y. (2003). Rationalism and inconsistency between predicted experience and decision. *Journal of Behavioral Decision Making, 16*, 257-272.

Hsu, C. L., & Lu, H. P. (2004). Why do people play on-line games? An extended TAM with social influences and flow experience. *Information and Management, 41*(7), 853-868.

Kano, N., Seraku, N., Takahashi, F., & Tsuji, S. (1984). Attractive quality and must-be quality. *Quality (Hinshitsu): The Journal of the Japanese Society for Quality Control, 14*(2), 39-48.

Kivetz, R., & Simonson, I. (2002). Self-control for the righteous: Toward a theory of precommitment to indulge. *Journal of Consumer Research, 29*, 199-217.

Leclere, F., Schmitt, B. H., & Laurette, D. (1994). Foreign branding and its effects on product perceptions and attitudes. *Journal of Marketing Research, 31*, 263-70.

Laurent, P., & Cleland-Huang, J. (2009). Lessons learned from open source projects for facilitating online requirements processes. In M. Glinz & P. Heymands (Eds.), *Requirements engineering: Foundation for software quality* (LNCS 5512, pp. 240-255). Berlin: Springer.

Leffingwell, D., & Widrig, D. (2003). *Managing software requirements: A use case approach* (2nd ed.). Boston, MA: Addison-Wesley.

Leonard-Barton, D. (1995). *Wellsprings of knowledge: Building and sustaining the sources of innovation*. Boston, MA: Harvard Business Press.

Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market driven software product development. *Software Process Improvement and Practice, 11*(1), 7-19.

Li, X., & Hsieh, P. A. (2007). Impact of transformational leadership on system exploration in the mandatory organizational context. In *Proceedings of the International Conference on Information Systems* (pp. 1-20).

Lindgaard, G., Fernandes, G., Dudek, C., & Brown, J. (2006). Attention Web designers: You have 50 milliseconds to make a good first impression. *Behavior & Information Technology, 25*(2), 115-126.

Liu, S. H., Liao, H. L., & Pratt, J. A. (2009). Impact of media richness and flow on e-learning technology acceptance. *Computers and Education, 52*(3), 599-607.

Mano, H., & Oliver R. L. (1993). Assessing the dimensionality and structure of consumption experience: Evaluation, feeling, and satisfaction. *Journal of Consumer Research, 20*, 451-466.

Matzler, K., & Sauerwein, E. (2002). The factor structure of customer satisfaction: An empirical test of the importance grid and the penalty-reward-contrast analysis. *International Journal of Service Industry Management, 13*(4), 314-332.

Mikulic, J., & Prebez, D. (2011). A critical review of techniques for classifying quality attributes in the Kano model. *Managing Service Quality, 21*(1), 46-66.

Mittal, V., Ross, W., & Baldasare, P. (1998). The asymmetric impact of negative and positive attribute-level performance on overall satisfaction and repurchase intentions. *Journal of Marketing, 62*, 33-47.

Oliver, R. L. (1997). *Customer satisfaction: A behavioral perspective on the consumer*. New York: McGraw-Hill.

Pomeroy, I. M., Clark, C. R., & Philip, I. (2001). The effectiveness of very short scales for depressions screening in elderly medical patients. *International Journal of GeriatricPsychiatry, 16*(3), 321-326.

Racheva, Z., Daneva, M., & Buglione, L. (2008). Supporting the dynamic reprioritization of requirements in agile development of software products. In *Proceedings of the Second International Workshop on Software Product Management* (pp. 49-58).

Robins, R. W., Hendin, H. M., & Trzesniewski, K. H. (2001). Measuring global self-esteem: Construct validation of a single-item measure and the Rosenberg self-esteem scale. *Personality and Social Psychology Bulletin*, *27*(2), 151-161.

Rothwell, R. (1976). Marketing: A success factor in industrial innovation. *Management Decision, 14*(1), 43-53.

Saade, R. G. (2007). Dimensions of perceived usefulness: Toward enhanced assessment. *Decision Sciences Journal of Innovative Education, 5*(2), 289-310.

Saaty, T. (1980). *The analytic hierarchy process: Planning, priority setting, resource, allocation*. New York: McGraw-Hill.

Sawhney, M., Verona, G., & Prandelli, E. (2005). Collaborating to create: The Internet as a platform for customer engagement in product innovation. *Journal of Interactive Marketing, 19*(4), 4-17.

Schmitt, B. H., & Simonson A. (1997). *Marketing aesthetics: The strategic management of brands.* New York: Free Press.

Sharma, R., Yetton, P., & Crawford, J. (2009). Estimating the effect of common method variance: The method-method pair technique with an illustration from TAM research. *MIS Quarterly, 33*(3), 473-490.

Sheehe, P. R., & Bross, I. D. J. (1961). Latin squares to balance immediate residual and other effects. *Biometrics, 17*, 405-414.

Sillitti, A., & Succi, G. (2006). *Requirements engineering for agile methods.* In A. Aurum & C. Wohlin (Eds.,), *Engineering and managing software requirements* (pp. 309-326). Berlin: Springer.

Strahilevitz, M., & Myers J. G. (1998). Donations to charity as purchase incentives: How well they work may depend on what you are trying to sell. *Journal of Consumer Research, 24*(4), 434–46.

Vavra, T. G. (1997). Improving your measurement of customer satisfaction: A guide to creating, conducting, analyzing and reporting customer satisfaction measurement program. Milwaukee, WI: ASQC Quality Press.

Venkatesh, V. (1999). Creation of favorable user perceptions: Exploring the role of intrinsic motivation. *MIS Quarterly, 23*(2), 239-260.

Venkatesh, V., & Speier, C. (1999). Computer technology training in the workplace: A longitudinal investigation of the effect of mood. *Organizational Behavior and Human Decision Processes, 79*(1), 1-28.

Venkatesh, V., Speier, C., & Morris, M. G. (2002) User acceptance enablers in individual decision making about technology: Toward an integrated model. *Decision Sciences, 33*(2), 297-316.

Veryzer, R. (1995). The place of product design and aesthetics in consumer research. In F. R. Kardes & M. Sujan (Eds.), *Advances in consumer research* (vol. 22, pp. 641-645). Provo, UT: Association for Consumer Research.

Von Hippel, E. (1988). The sources of innovation. New York, NY: Oxford University Press.

Von Hippel, E., Ogawa, S., & de Jong, P. J. (2011). The age of the consumer-innovator. *Susumu Ogawa*, *53*(1), 27–35.

Wiegers, K. E. (1999). Automating requirements management. *Software Development, 7*(7), S1-S6.

Witell, L., Löfgren, M., & Gustafsson, A. (2011). Identifying ideas of attractive quality in the innovation process. *The TQM Journal, 23*(1), 87-99.

Zhang, P., & von Dran, G. M. (2002). User expectations and rankings of quality factors in different web site domains. *International Journal of Electronic Commerce*, *6*(2), 9-34.

About the Authors

Adarsh Kumar Kakar has a PhD in Management Science with a research interest in elicitation, capture and processing of various types of users' information system requirements. He has over three decades of experience in the software industry and has worked as consultant for many Fortune 500 companies. He is current working as an Assistant Professor in the department of Computer Information Systems at Alabama State University. He has published in journals such as *Information and Software Technology, Computers in Human Behavior, Journal of Computer Information Systems* and *Journal of Decision Systems*.

Transactions on Human – Computer Interaction