17

Systematic creativity: a methodology for integrating user, market and engineering requirements for product definition, design and usability testing

A. C. Salvador & J. C. Scholtz
Intel Corporation
5200 N.E. Elam Young Parkway, Hillsboro, Oregon, USA 97124
phone: 503.264.6455
fax: 503.264.0081
e-mail: tony_salvador@ccm.jf.intel.com

Abstract

The rate of emerging technological innovations has exceeded the rate at which individuals can assimilate new technologies into their lives. Yet, businesses continue a breakneck pace for the development of new technologies resulting in the creation of products based on a particular type of technology, i.e., technocentric product development, rather than on its utility and usability, i.e., user centric product development. One reason for this misplaced focus on technology rather than on the intended users, may be that few methodologies exist to genuinely *incorporate* the user in the product development cycle, which would otherwise more tightly link users' requirements to product design and usability testing. This is especially the case for software user interface development. This paper identifies a framework of a discrete number of specific data classes fundamental to incorporating the user into product development.

These data classes not only represent the user, but also foster a rational decision making process for product definition, design and usability testing. This framework can be used independently of the requirements techniques used for acquiring the data from users and independently of the software development methodology used in the product development process, which implies that these general data class types are applicable to nearly all software development projects. The framework guides requirements data acquisition and facilitates requirements evaluation for systematically exploring alternative designs and testing those designs. The framework also provides necessary systematic guidance to the design process,

L. J. Bass et al. (eds.), *Engineering for Human-Computer Interaction*

while retaining the necessary creativity to design a novel product and marketable user interface. Finally, the simplicity and practicality of the framework makes it attractive for use in real product development. An actual example is included to illustrate the main points of the paper.

Keywords
User requirements, product definition, product design, usability testing

1 INTRODUCTION

For a product to be successful, customers and users must clearly perceive the value of the product where a product's value is based on it's utility and it's usability. These two characteristics are intrinsically intertwined, yet there are few methodologies that directly link user requirements in terms of utility and usability to application definition, design and usability testing. This paper proposes a methodology that captures user requirements in a single data set that can be stored, represented, accessed and manipulated to directly support product definition, design and testing.

Specifically, this paper has three goals. First, the paper presents a consistent and systematic framework for incorporating the user into product development by collecting, representing, accessing and managing user and product requirements to support product definition, design and user testing. Second, the framework accounts for many practical concerns of real product development, including the early identification of conflicting requirements, early and systematic engineering involvement, the ability to manage the complexity of the product and the ability to set a completeness standard. Third, the consistent data representation facilitates communication among marketing, design, testing and development folks on the product team by providing a commonly understood foundation of concepts required to define, design and test the product from a user's point of view. Although a description of this methodology is provided, complete examples of real products are not included in this paper.

The benefits of a methodology as offered in this paper are to provide a practical means to genuinely incorporate users into product development. One assumption we do make is that early user input is critical to most successful products. It is equally important to remain competitive, which implies maintaining the ever increasing rate of application development, adhering to seemingly arbitrary (marketing) schedules and accounting for technological constraints. Furthermore, while businesses do indeed require products to be produced on time, the products must also be truly innovative, useful, usable and are accessible, that is, people can incorporate the product into their daily activities. However, with the technology boom, market niches for new applications must continue to decrease in terms of relative size and number (Moore, 1991), resulting in a successful product being an ever smaller needle in an increasingly larger haystack.

Identifying the functions that a product should support requires a systematic exploration of daily life to understand how assorted product concepts and functions might fare. However, this is not to diminish the role of creativity necessary for product design. Combining, adding and eliminating requirements effectively is a creative process, as is the product design itself, albeit practically constrained and guided by the requirements themselves. Thus, the challenge to human factors is to find methodologies that can be used to systematically search for change while retaining the creativity necessary for developing products.

Any methodology to meet this challenge should meet at least five criteria. First, product definition cannot be carried out iteratively with users due to constrains on both the users and developers. Second, any user requirements methodology must be eminently adaptable to whatever software development methodology is in place in a particular organization. Third, to be most effective, the requirements data collected must be represented in a single format and applicable to as many phases of the product development as possible. Fourth, the data set and the methodology must be sustainable, i.e., it should be possible to augment the data set when considering subsequent projects. Fifth, the data set must be accessible not only for product definition, but also to guide design and development, thus retaining the user throughout the development process.

2 PREVIOUS METHODS

This paper proposes a methodology that derives it's essential elements from prior work in process control, but that modifies the methodology appropriately for the technical and practical needs of retail products. The current methodology is based on two concepts that were originally combined in the Functional Information and Knowledge Acquisition (FIKA) modeling approach (Sundstrom, in press, Sundstrom & Johanssen, 1989). According to the FIKA methodology, a viable requirements methodology should first determine the data that need to be collected in the field and, second provide a structure sufficient for integrating data from the field to determine what the support system should do, which links the requirements to the system's behavior (Sundstrom, in press). Additionally, the FIKA modeling approach to capture user requirements is a way that reflects what the user needs the system to do. Logically, then, the presentation layer design can be derived from the FIKA model as demonstrated in Salvador & Sundstrom (1993). However, the FIKA methodology is suited specifically for large process control systems which are designed specifically for monitoring, fault detection, diagnosis, compensation and correction. These activities, *per se*, do not relate to most retail products.

On the other hand, methods more applicable to retail products focus exclusively on requirements elicitation and product definition. For example, methodologies, such as PICTIVE (Muller, 1993) and Contextual Inquiry (Holtzblatt & Jones, 1993) have been developed to fill this need. However, both PICTIVE and Contextual Inquiry focus nearly exclusively on eliciting and manipulating requirements for product definition, but are less well suited for guiding design and usability testing (See Appendix A). Thus, there is no single existing framework for eliciting and organizing user requirements in a format applicable to product definition, design and usability testing.

The current methodology builds on some basic concepts first outlined in the FIKA model, and extends the methodology specifically for retail products -- structuring user requirements appropriately for product definition, design and usability testing. The remainder of this paper describes the proposed methodology and provides explicit examples of how the framework can be used to support definition, design and user testing. First, we describe the Systematic Creativity framework. Then we provide a modified transcript of an actual interview which is then represented as specified by the framework. Finally, we identify the benefits of using this framework for definition, design and usability testing.

3 DESCRIPTION OF FRAMEWORK

This framework divides the user interface design task into two parts: first, identifying what functions the product must consider and second, guiding the design of how the product will provide those functions to the users. The first part refers to the collection and representation of user and product requirements as pertains to product definition. The second part refers to the ability to simply manipulate the requirements data to guide design and testing. Of course, it remains the province of the user interface designer to actually generate user interface controls, widgets, metaphors, etc., reflecting the creativity of the product designer, while the framework retains the systematicity related to what it is that is actually being built.

We provide an example in Section 3 that attempts to show how the different parts of the framework fit together. The example, although an actual excerpt from an actual interview, is for illustrative purposes only and is not intended to convince or confirm the utility of this framework. In addition, Section 4 describes how the methodology can be used in product definition, design and usability testing. It may be useful to refer to either or both of these sections as you read the following description.

3.1 Framework

The framework in Figure 1 illustrates the high level view of the data classes that comprise the framework and their interrelationships. Each cell in the model is related either directly or indirectly to other cells in the model, thus providing continuity and context from the user goals through to specific objects and actions from the product definition phase through the design and usability testing phases.

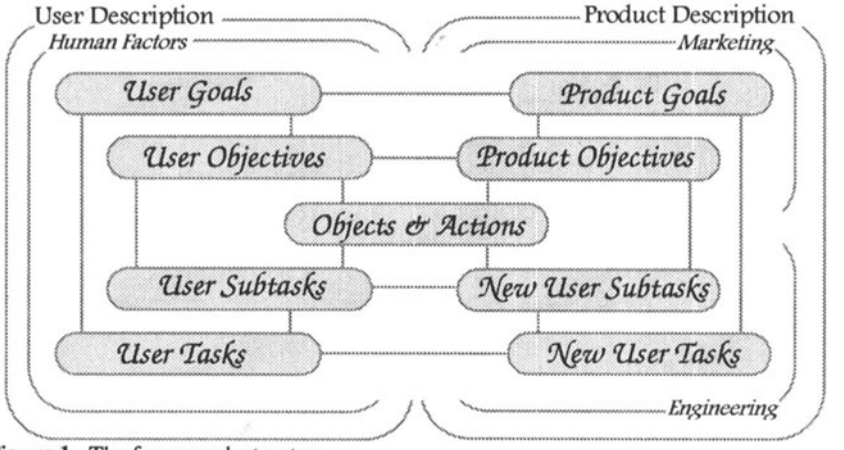

Figure 1 The framework structure

The top half of the figure represents goals and objectives related to both the users and the product itself. The bottom half of the figure represents the tasks that users and/or applications need to perform to meet the user and product goals. The left side represents data classes indicative of the user, obtained during requirements elicitation. The top left quadrant

represent the first level user goals and objects that will be used to determine product definition and design direction. The lower left quadrant represents the tasks that users currently perform to support their goals. The right side represents data classes related to the product, *per se*. The upper right quadrant represents data contributed through traditional marketing input. The lower right quadrant represents the recombinations of objects and actions, typically generated by user interface designers and engineering, relative to the new application technology. The center reflects the objects and actions abstracted from the context of the goals and tasks. It is important to recognize that the outer edges of the framework maintain the context of the abstract object and action classes in the center of the figure. Similarly, moving from left to right represents a shift from data representing a set of users performing tasks in support of specific user goals to data representing a set of users performing tasks in support of specific user and product goals using the product.

In addition, each cell in the network may have associated with it a set of one or more facilitators, a set of one or more obstacles or both. Facilitators are items or processes that assist the user's achievement of the user goal, objective, task or subtask. Obstacles are items or processes that hinder the user's achievement of the user goal, objective, task or subtask. There are three reasons for this structure. First, in practice, no modeling framework is ever complete. Thus, facilitators and obstacles provide the necessary flexibility to add additional bits of information as related to the main data classes in the framework. By viewing these additional data items as facilitators or obstacles, additional contextual information is immediately identified during the interview, and is directly applicable to the design of the product. Second, one often finds that users are either faced with particularly hindering obstacles, such that if the obstacle were removed, their tasks would be sufficiently improved and their goal more efficiently or effectively attained. Third, many users exhibit tremendous ingenuity when faced with particular obstacles creating local improvements that greatly facilitate their efforts. These data items are related to specific cells and are described further below.

This framework maintains the separation of user goals from product goals for several reasons. First, product goals (from marketing) often conflict with user goals (derived from requirements analysis). Second, to the extent that product and user goals correspond, one may be able to predict the perceived utility of the product. Third, depending on how the product is instantiated, one may be able to make a statement about the ability of the user population to incorporate the technology into their daily activities. Fourth maintaining the separation allows for targeted, differential user testing, streamlining the user testing process, saving time and money. Fifth, product goals can be used to determine the completeness of the user goals and vice versa, each serving to augment the other.

3.2 User goals

User goals refer to the user's *first level description* of the desire to change or maintain the current state of their activities. Figure 2 shows the expanded view data structure for user goals. *First level description* refers to the fact that user goals are not recursive. All user goals occupy the same heterarchical level; that is, there is not a goal hierarchy or any implied order. It is also this level of goal description that most significantly impacts product definition and design and that can be used to design and implement usability tests. One can, however, order, rank or otherwise organize goals and their associated data classes according to some imposed superordinate context. For example, user goals may be ordered with respect to a particular

product goal or set of product goals within the context of the current product being developed or in the context of a related or separate product. This is described more completely in the section *Using the Framework for Product Definition.*

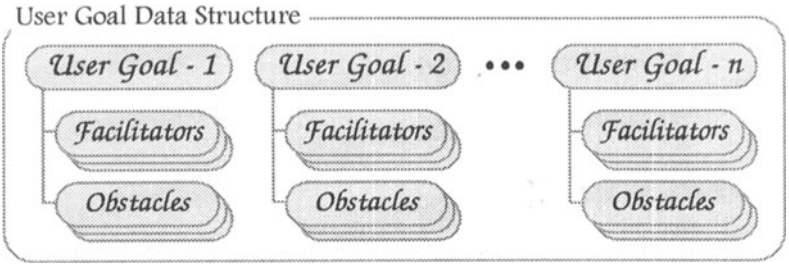

Figure 2 User goal data structure.

User goals may be determined to be complete as defined in terms of a particular user by exhaustively probing various characteristics of the work environment working from core activities, i.e., those most closely related to the product under development, toward more tangentially related activities. Appendix A suggests interview techniques that tend to improve the completeness of the user goal set for each interview.

A set of user goals can also be determined to be complete for a particular product. In this case, there are two measures. First, over time, the number of new user goals discovered during successive interviews will diminish. The number of interviews required to reach this point of diminishing returns depends on the breadth of the user population and the product being developed. Logically, this seems very similar to what usability research has determined regarding the number of subjects required in a usability test to identify the number of problems found (cf., Nielsen & Landauer, 1993). Although we have not conducted a study on this particular issue, it has been our experience that the number of newly identified goals diminish within 4-6 users, where the users were of a similar type, or where inter-user differences relative to the particular topic were very small.

As mentioned previously, the framework not only captures information, but guides the collection of information. For example, by specifying that user goals were to be collected, the interviewer is required to model the interview in terms of goals, which are apparent only if the interviewer is looking for them. There are at least two ways that using this framework guides the data collection process relative to user goals. First, the product goals and objects should limit the scope of the interview to a particular aspect of the user's work environment. Second, the user goals derived from each interview can be used to guide and refine subsequent interviews.

3.3 User objectives

Product objectives describe why the user goal exists, i.e., why the user wants to meet that goal. Figure 3 shows an expanded view of the user objectives data class. As with user goals, facilitators and obstacles may also be associated with the objective. Understanding why a user needs to meet a particular goal provides information related to the relative importance of the goal, and suggests specific target areas for the product to improve.

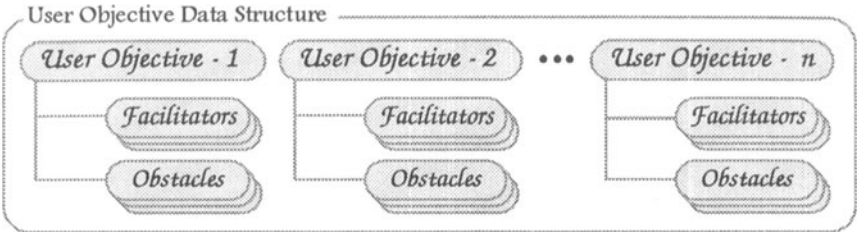

Figure 3 User objective data structure

Figure 4 shows the possible relationships of user objectives to user goals. Note that most often user objectives are directly related to particular user goals, one to one, simply because if done properly, a range of user goals will each be achieved for a range of reasons, i.e., anticipated gains. However, it is not completely unreasonable for some user objectives to be shared by more than one user goal. For example, a user goal may have more than one user objective associated with it increasing the chance that another user goal may share a similar objective. One expects this to occur more frequently with more complex products. In addition, facilitators and obstacles may be common to more than one user goal. User objectives cannot exist independently of at least one user goal.

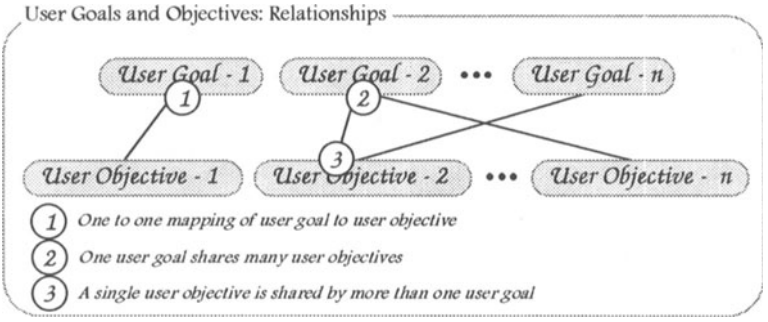

Figure 4 User goals and objectives possible relationships.

3.4 User tasks and sub-tasks

User tasks and subtasks describe what the user does to achieve their user goals within the constraints of their objectives. Tasks are represented as high level object-action pairs, and subtasks are represented as *primitive* object-action pairs, one object is paired with a single action applied to that object. Primitive refers to the smallest meaningful (to the user) object and action. For example, in an address book application, "surname" would be a primitive object; "erase" may be a primitive action applied to that object. "Erase" may also be applied to other primitive objects. An object-action pair is the primitive behavioral unit. As with user goals and objectives, there is no implicit or explicit hierarchy; for example, tasks themselves are not intrinsically hierarchical and neither are subtasks. However, subtasks are hierarchical relative to task, just as objectives are related hierarchically to goals.

Figure 5 shows an expanded view of the task and sub-task data structure. First, each object can be clearly attached to more than one action. Similarly, actions can apply to more than one object. Second, as with goals and objectives, facilitators and obstacles are also attached. Typical facilitators for tasks and subtasks might include who actually performs the task or subtask (e.g., the user or a secretary), or the media on which the objects reside (e.g., on paper or electronically). Third, attributes of a particular object or action are associated with that object.

When viewed from the perspective of a task, individual subtasks may be linked and ordered in the context of that particular task representation. Similarly, tasks may be linked and ordered when viewed from the perspective of the user goal. Therefore, within each data class, there are no explicit or implicit dependencies. If there is a hierarchical relationship in the task category, it is probably necessary to generate a user goal that removes this relationship.

This characteristic of hierarchy free data classes is one of the main strengths of the framework. The cleaner the data structures within each class, the easier the structure is to use during product definition, design and user testing, as will be described below. From a practical perspective, maintaining homogeneity within a data class makes the interviewer's task easier in terms of maintaining clarity during the interview. Within class homogeneity makes the results of the requirements analysis considerably more accessible to others within the product team. Also, others can add to the database, with relative ease by simply adding goals, tasks, without having to worry about combinatorial interrelationships. Thus, the database is more easily maintained than modeling structures geared for developing expert systems (e.g., Sundstrom, 1991, Mitchell & Miller, 1986)

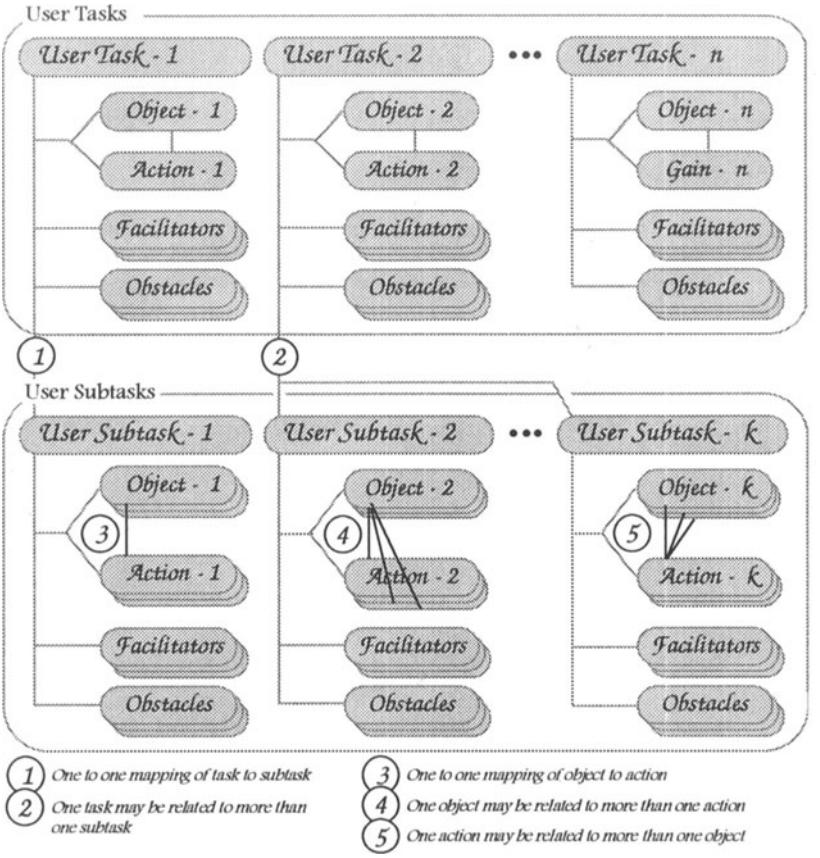

Figure 5 User task and subtask data structures

3.5 Product goals

Product goals are defined as *product characteristics* that make it compelling to the potential *buyer, or customer*. This structure is identical to the user goal structure, including the inclusion of facilitators and obstacles. The difference between user goals and product goals is contained in the content of the data class -- not it's structure. There are three distinct differences between the type of information captured in product goals as compared to user goals. First, product goals consider *product characteristics* derived from the user population in the aggregate, which is closely related to traditional marketing data collected prior to product definition. An example of a product goal might be *the product must be portable*; a corresponding user goal might be *access to information at any time*. Second, product goals must account for customer requirements, as opposed to users. For example, in large corporate settings, the Management Information Systems (MIS) department would most likely

be responsible for *purchasing* the product, so their concerns must be met. However, they may not be the principle users of the purchased product. Third, product goals represent the product in the context of the market rather than in the context of the user. That is, product goals consider aggregate market populations as well as external market forces, such as competitive pressures, pricing pressures, standards, timing factors, etc. The result is that marketing will provide some input that complements user goals and objectives but that also expands the total context in which the product must be defined, designed and tested.

Unfortunately, sometimes, product goals conflict with user goals. For example, a product goal might suggest that the purchaser desires a certain amount of control over use of the product over the company's network, while users might clearly desire unlimited access. Together, product goals and user goals determine why particular features are included or not included in the product and which features are considered more important. This is critical to systematically guide design and usability testing. For example, including a particular capability simply because competitors have this capability may be necessary for competitive purposes, but perhaps does not meet an important user goal and therefore, does not have to be as prominent in the interface design as other features.

Finally, the combination of both product goals and user goals suggests areas for further investigation, as marketing research may have discovered a particular item that was not revealed through a user requirements analysis and vice versa. Thus, by retaining the structure of the data classes, comparisons across data sources can be made, further guiding a systematic data collection process.

3.6 Product objectives

Product objectives qualify the product goal in terms of why the product should achieve a particular product goal. The class structure for product objectives is identical to that of user objectives, including facilitators and obstacles, and the differences between the two data classes are determined by the content. Obstacles refer to the temporal, economic or personnel reasons users/customers may not purchase the product. Facilitators refer to why the user might consider purchase of the product. Product objectives and their facilitators and obstacles differ from user objectives in that the content of product objectives reflects anticipated user behaviors weighed against other anticipated behaviors unassociated with the product being discussed. Also, the content associated with facilitators and obstacles will also differ considerably because product objectives pertain to the context of anticipated events and the user objectives pertain to the context of current events. Relationships between product goals and objectives are similar to those described for user goals and objectives.

3.7 New tasks and new subtasks

New user tasks and new user subtasks describe the intended tasks and subtasks users will need to perform to achieve their goals given a functional description of the new application. The class structure is identical to that of user tasks and subtasks. New tasks and subtasks therefore, are derived from the product definition and design, from which the functional description of the new product is generated. That is, the new tasks and subtasks reflect the total object-action data class organized to support the context specified by the user and product goals and objectives. (See next section.)

Using this framework, new tasks and subtasks are derived by considering user and product goals and objectives as well as considering the object action pairs data class. Completeness is determined by sufficiency in meeting user and product goals. The quality of the functional design is determined by how well the design meets or exceeds user and product objectives.

3.8 Objects and actions

Figure 6 shows the expanded object-action cell. This cell represents a class structure fully free of context. All objects and actions identified in user tasks and subtasks are collapsed showing the minimal set necessary to represent each object and each action exactly once, with associated relationships. Additional object action pairs identified during design can also appear in this cell as required by the new product. This cell, therefore, represents the building blocks necessary to construct the new product, in the absence of any particular context. Consequently, from the user perspective, this cell provides an effective set of minimal requirements software developers need to begin constructing an underlying set of capabilities that can be combined in the appropriate context at the presentation level.

New Subtask and New Task Development also contributes to the object action class, as shown in Figure 7. First, they may contribute new objects and actions that become required during development. Second, and more specifically, new tasks and new subtasks will require some type of *initiators* associated with actions and the resultant feedback related to that action is represented as *states* of the object. Thus, each action has associated with it one or more ways to initiate that action, and these are an integral part of the product.

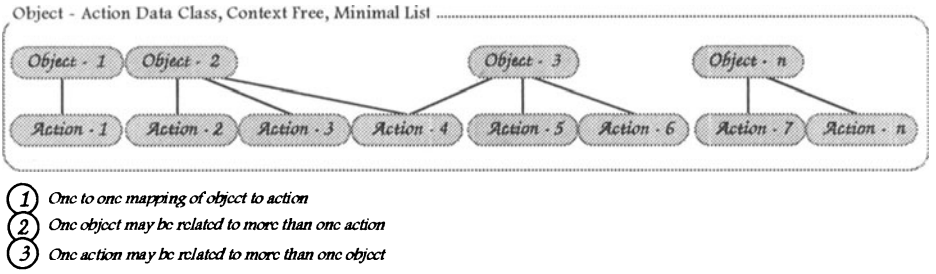

Figure 6 Object-Action data relationships.

Similarly, objects acted upon by certain actions can be described by one or states describing the status of that object. For example, while writing this document, the author can initiate a "save" action through the keyboard, the menu, or automatically. The document object can be in either an unsaved state or a saved state depending on the recency of the action.

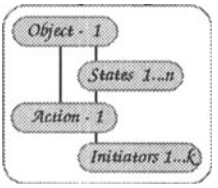

Figure 7 Object-action data class structure with object states and action initiators.

Finally, a set of objects and actions can be derived from the new tasks and subtasks. For example, register product, enter name, enter address and send form. As the product gets designed, it becomes clear that the initiator for the product registration is the system and that the feedback for entering the name is seeing the name appear in the appropriate field, and so on.

4 EXAMPLE

The following is a partial transcript of an actual interview, which we will use as an example illustrating the various data structures just described. The purpose of the interview was to determine how Allen keeps track of the people he knows, i.e., how he manages names, addresses, phone numbers, etc., with the supposed intention of building some kind of product in this area.

Investigator:	Allen, tell me about how you keep track of your friends and family. Specifically, how do you manage, organize and get at their addresses, phone numbers, etc.
Allen::	Well, I have a little black book that I keep addresses and phone numbers in -- mostly phone numbers.
Investigator:	Is that the only thing you use.
Allen::	Yes.
Investigator:	What about at the office?
Allen::	I've only been working for about 6 months, so I don't have many work related contacts. But I do have phone numbers of the people I work with on yellow stickies on my computer.
Investigator:	Don't you carry your black book with you?
Allen::	No. I generally leave that at home.
Investigator:	Oh. Are yellow stickies sufficient for keeping track of the people you work with?
Allen::	Yes, I only have a few numbers right now. I suppose if I had many, I'd have to do something else.
Investigator:	Do you always leave your little black book at home?
Allen::	No, I take it when I travel.
Investigator:	Why?
Allen::	To send postcards, mostly.

Based on the transcript excerpt of our interview with Allen, we identified four user goals: 1) maintain black book currency, 2) separate business & personal addresses, 3) have ready access to frequently used numbers at work and 4) access personal addresses anywhere at anytime. These goals were derived during the interview based on Allen's responses. The interviewer periodically repeated the list of goals to Allen, who commented on whether they were accurate.

Considering one of Allen's user goals, access personal address information anywhere, we identified two objectives of this goal: 1) to send postcards and 2) to make calls in an emergency. Since Allen decides first, to whom he will send a postcard, a facilitator for sending

postcards was that the book was sorted alphabetically by name. An obstacle was that Allen can only alphabetize conveniently by the first letter of the name; thus, within letter, the names are no longer alphabetized. Note that the facilitator referred to the objective of sending a postcard and that sending a postcard referred to the goal of accessing personal address information anywhere; the same is true for the obstacle. Why Allen wants to send a postcard is not considered in this structure.

A task related to this user goal is to find the address of a friend, Cathy Smythe, to whom he wants to send the postcard. Subtasks include opening the book, finding the "S" tab, and scanning the S list for Cathy Smythe's name, and so on. Note that this task and these subtasks are probably related to more than one user goal and/or objectives.

From each task and subtasks we derive the object action pairs. Considering only the task and subtasks mentioned above there are four object action pairs: find name, find address, open book and scan list.

Now, examining the product side of the framework, an example product goal is to collect product registrations, which has little to do with the user, at least from the user's point of view. That is, the marketing effort will probably aim at finding ways of inducing users to register their purchase. Because registering the product is not perceived by users to be a valuable goal, and because registering takes time and distracts the user from their goals using the product, this product goal conflicts to some extent with the user goal. Goal conflicts suggest features that users do not perceive as useful, therefore, a design which allows users to accomplish a product goal with little effort is highly desirable.

Two objectives associated with the product goal of collecting product registrations are compile a mailing list and keep accurate records for updates. Finally, new tasks for the user might include entering their own address information and sending that information to the company, both of which have associated subtasks. The objects and actions from these new tasks and subtasks would contribute to the objects and actions data class, such as enter first name, enter last name, print form, mail form, etc.

In this example, registering the product is a new task for the user. The user will have to register the product by completing the following subtasks: entering information such as their name and address information and sending the completed form to the appropriate destination.

5 APPLYING THE METHODOLOGY TO PRODUCT DEVELOPMENT

As mentioned, the data captured by this framework can be used most effectively for product definition, design and usability testing. In each of these three phases the data classes and the data within each class can be manipulated to optimally support the activities in that particular phase through the use of some relatively simplistic sort and filter routines. That is, in each phase, the same data may apply to very different sets of issues and product decisions; each data class may also be more or less important relative to each phase. The next three sections describe specifically how this framework can be used to guide each of these three product development phases.

Applying the Framework to Product Definition

This framework can be used to guide the definition of the product in three ways. First, the framework supports the systematic collection of user data relevant to product definition. Goals and objectives specify what the user needs to achieve and why he/she wants to achieve it. The

associated facilitators clearly identify those characteristics of their current overall environment conducive to achieving their goals and the obstacles clearly indicate where their current environment is less well suited to achieve the necessary goals. Tasks and subtasks gather the information objects and actions necessary to understand in detail the requirements associated with particular goal contexts.

In addition to the systematic collection, this framework is useful during the elicitation process (Sundstrom & Salvador, in press). Using this framework to represent the data, the interview effort to collect this level of detail is relatively minimal, requiring about 20-30 minutes of the users time to collect user goals and about 60 to 90 minutes to collect objectives, tasks and subtasks. The time can be further reduced on subsequent interviews as data collected from one interview can also be used to augment further interviews by avoiding unprofitable dialog and more quickly exploring the user's space. By specifying what data to collect, interviews are not limited to any one person or team, i.e., any person (who must be a skilled interviewer) can contribute accurately to the data set, as well as rely on the existing data set to enhance their own interviews. Any person on the development team can quickly access the data set at any time to understand the current state of the data and to begin making inferences based on the data as necessary to provide input to marketing or engineering as necessary.

A second major strength of this framework is the ability to systematically manipulate the collected data to clearly define what the product will do by identifying what user and product goals the intended product will support. The point of this exercise is to determine one or more constellations of goals the product will support. There are several ways to manipulate the data to construct assorted constellations; here are two simple examples. One way is simply to sort the data according to obstacles and their associated goals and objectives, thus identifying what goals are currently difficult for users to achieve. Another way is to qualify the user goals and objectives in some fashion. For example, it is possible to collect rankings and/or ratings of importance, suggesting an ordered list of what the product should do. It is also possible to assign relatedness scores to the data items and conduct cluster or factor analyses.

Also, more sophisticated, second order analyses, can be conducted. For example, cross tabulations can be run to garner a much greater amount of information about the data set. For example, it is simple to determine the concordance between goals (as defined by obstacles) and goals as rated by importance. More importantly, the results of the cross tabulation (what needs to get done and what's hard about it) are not only information intensive, but also easy to understand. Furthermore, such analyses can easily be conducted for any combination of data looking for correlations, redundancies, etc.

Another way to create goal constellations to define the product is to be less formal, but nonetheless systematic. For example, it would be quite possible to construct a continuum of scenarios by considering one or more user goals and their associated data. One might wish to examine a potential product by considering all data associated with n goals in the form of a large inclusive scenario. It is equally feasible to construct small scenario vignettes. For example, one could create a two sentence scenario as related to a particular user goal or the same length scenario as related to a particular sub-task. The granularity of scenario is systematically determined by the data class item that contributes the content of the scenario. Similarly, the decisions one wants to make about the product definition should correspond to the scenario and its granularity.

Third, one can use this framework to identify functionality tradeoffs based on any criteria that can be used to further qualify the data, e.g., cost to develop, development time, resources required, etc. By viewing the data in this framework can crisply delineate areas for further investigation, including augmented requirements collection and testing to decide among product goals. Product extensibility in terms of user and product goals can potentially be identified early in the development process. The user and product objectives can be use to determine the set of high level assumptions related to the product design that guide the implementation of the architecture and user interface. Conflicts between product and user goals and objectives (i.e., marketing and design) can be settled earlier in the development process. Additionally, engineers can begin thinking of engineering solutions for particular goals or goal sets, thus emphasizing their creativity regarding how to instantiate a particular function that is within the boundaries of the product definition.

5.1 Applying the framework to product design

The previous section described how one might use this framework to define *what* the product would do. This section describes using the framework to guide the process of designing *how* the product will perform the specified functions. The framework supports two main design activities: developing the user interface, specifically through the creation of new tasks and new subtasks, and managing the complexity of the product from the user interface perspective. In addition, there are some practical organizational benefits that derive from use of this framework.

The goals and objectives represented in this framework provide specific guidance for identifying a) what types of controls to develop and b) how to organize controls to form tasks and subtasks that best support user goals. To design the product, engineers use two kinds of user interface controls: standard controls germane to the operating system under which they are developing and custom controls, which are unique and need to be developed by the engineer. The controls represent action initiators. Typically, engineers assign specific actions to these controls and associate the particular control-action combination to a particular object. Object-action pairs are then sequenced together to create subtasks and tasks.

Designers can also consider their product design based on parametric qualifications of user and product goals. For example, considering only the object action pairs, designers would randomly organize the user interface to accommodate all subtasks and tasks that they generate. However, the organization of the controls in the user interface can be determined by considering alternative views of the user and product goals. For example, user goals can be characterized by importance and/or frequency which can indicate the level of particular subtasks and tasks within the user interface. Another example is to qualify goals, tasks and subtasks as pertaining to novice, intermediate and advanced users. Additionally, given projections about the intended usability of the product, usability objectives can be associated with particular user goals, such as learnability, efficiency, memorability, error proneness and user satisfaction. The designer is able to organize goals and objectives by facilitators and obstacles, making it easier to track benefits of the product in the current design. The fact that it is easy to organize and reorganize the data contained in the matrix makes it feasible to use this framework for design.

A second major benefit of using this framework for design is that the framework facilitates consideration of all objects related to specific actions for which a control is being designed. As described earlier, actions and objects are related to each other in one of the following ways:

one to one, one to many or many to one. As the complexity of the product increases, the probability that the same action applies to many objects increases. When designing the control for a particular action, the context should not be a single object or a couple of objects, but all the objects related to that action. This framework makes it much easier to track the all objects associated with any one action and vice versa. Thus, when considering the user interface design in the context of one user goal, the designer can cross reference related goals based on associated object-action pairs. This allows the designer to not only consider the context of the user perspective during design, but can also consider the practical technology perspective necessary for generating a consistent, implementable architecture.

Finally, there are some down to earth practical benefits of using this framework to support user interface design. First, the framework clearly communicates the important requirements to which designers must attend. Whether the information remains in the framework representation or is transformed, e.g., into scenarios, designers can recognize the product requirements. Second, designers can check their designs for completeness as compared to the goals and objectives, facilitators and obstacles represented in the framework. Third, the framework provides a means for understanding how the product would possibly add functionality in future versions. In most cases, the product definition results from a paring of all user and product requirements. Designers can examine those user and product goals that are not supported in the product to establish how the product might extent to incorporate these features. Fourth, this framework complements object-oriented analysis in that the object-action data class specified in this framework is a proper subset of the information needed for a complete object-oriented analysis and complements the work of the analyst who will produce the complete set of domain objects needed. Finally, using this framework supports the maintainability of the user interface during design and implementation. Whenever there is a change, designers and engineers can refer to the framework to examine the ramifications of one change on the rest of the product, e.g., to understand the impact of an engineering modification of a particular action, the engineer can review all action initiators, objects and object states before commencing the change.

5.2 Using the framework for usability testing

The third major focus of product development supported by use of this framework is usability testing. The framework supports identification of issues related to usability testing as well as providing usability test metrics free from the traditional speed and accuracy dependent variables. As with definition and design, there are also some practical benefits from an organizational perspective.

5.2.1 Usability testing: new tasks and subtasks related to user goals v. product goals

Usability testing in a fast paced product development poses two difficulties for the development process. The first is identifying what to test and the second is identifying when to test it. The framework allows an early identification of what issues warrant usability tests and what issues do not warrant a test, e.g., in the definition and design phases. For example, one can test new tasks relative to supporting a user or product goal, test for the ability of the product to support one or more high priority user goals and test high priority tasks in the

context of one or more user goals. Thus, the human factors practitioner can identify critical usability issues early, perhaps modifying the development schedule or at least adapting to the development schedule, making plans and preparations in advance, facilitating the incorporation of usability testing in the product development cycle with a greater expectation and probability that usability results will impact the product. The designers can also use the framework to carve out product components that relate together either in a task or goal, knowing the product components required for particular tests. In addition, the user can track testing the content of testing schedules, decisions made, etc.

The framework also provides a structure by which to catalog usability test results for current or future reference. Consider a product currently under development: having tested the usability of a set of controls supporting a particular task, the object actions pairs that related to that test can be used to generalize to other tasks where those object action pairs exist. That is, the tester can sort the framework by that particular action and quickly identify other locations in the product that might require attention. For future reference, the framework can be used to identify previous usability results related to particular actions and objects and can then avoid those problems in the design of any particular product.

5.2.2 Usability metrics

Another distinct advantage of using this framework for testing is that the magnitude of a usability problem can be identified relative to the practical issues of how much time and how many resources will be required to remedy the problem and depending on when in the development cycle the problem was discovered. While it is clear that the difficulty of fixing a problem increases as the product nears later stages of development, what is not clear is that the difficulty of fixing certain problems, e.g., a missing, but desired, user goal, is most often much more difficult at any point in time than fixing a problem related to a particular subtask. For example, consider two sets of results. The first indicates that the feedback provided by the object state is insufficient for whatever reason, and the second indicates that the product is missing a key piece of functionality. Using this framework, it is clear that the first set of results has less impact on the product and product schedules than the second set. Thus, in engineering terms the magnitude of the test results can be qualified. It is also possible to weight the implications of test results from the user perspective. For example, if user goals and objectives have been qualified in terms of importance, results with implications for more important goals should have greater weight than results with implications for less important goals.

6 STRENGTHS AND WEAKNESSES OF CURRENT APPROACH

Newbery (1995) proposed five characteristics of any methodology to assess it's applicability in any particular project. These are: scaleability, efficiency, integrity & transferability, communicability, and effectiveness. The strengths and weaknesses of the current approach will be considered in this fashion.

6.1 Scaleability

Scaleability refers to the ability to shorten, cut, abbreviate and discard portions of the methodology without causing the whole process to become ineffective. The current methodology can be used effectively at any stage of development from pre-definition through to final usability testing, serving to organize and guide each phase of the development cycle. That is, it is linearly scaleable, so that the detrimental effects of eliminating involvement in any phase of development remains proportional.

6.2 Efficiency

The current product is quite efficient. In fact, if there is a weakness, it is in that the methodology fails to collect enough data. However, the purpose of this technique is to focus the development systematically on the product being developed. Other techniques, which are more suited for complete organizational descriptions, e.g., Contextual Inquiry, collect a wealth of information that is not particularly relevant for product design, per se, at the cost of time and resources. With Systematic Creativity, the practitioner chooses the domain of interviewing and analysis, (e.g. business communications), and all design activities concentrate on that area.

6.3 Integrity and transferability

Systematic Creativity attempts to integrate user, market and technological data systematically providing the context for design while allowing the creativity necessary to design new products. No other approach known to us and practical enough for retail products offers a single, consistent framework for collecting and using user, market and engineering requirements throughout the product development process. (But see Sundstrom, 1991, for this approach in process control.)

6.4 Communicability

Communicability refers to how effectively the methodology can communicate user work patterns to product team members? Again, the simplicity of the data classes, the straight forward approach to collecting data and the ability to demonstrate the interconnections among user, market and engineering needs provides for a convenient and systematic way to discuss the elements of a particular product.

6.5 Effectiveness

This characteristic refers to how well the methodology actually does incorporate user needs into the product definition and design. This technique focuses on including that information into the design that is most relevant to the user needs given the competing requirements from marketing and engineering. From that perspective the methodology fits well within the philosophy that there is no such thing as a perfect development environment, and so there is no such thing as a perfect product. However, there may well be superior techniques, e.g., Contextual Inquiry, for conducting a lot of focused work on a particular organization, e.g., for a custom product. Systematic Creativity is, however, a product of high-pressure commercial

software development, and so may be advantageous when the practitioners finds themselves brought in late, having too little time and resources to perform full user-centered design activities, and owning little political clout within the design team.

7 SUMMARY

We have described a methodology that has the potential to 1. provide a consistent and systematic framework for incorporating the user into product development by collecting, representing, accessing and managing user and product requirements to support product definition, design and user testing, 2. account for many real-life practical concerns including the early identification of conflicting requirements, early and systematic engineering involvement, the ability to manage the complexity of the product and the ability to set a completeness standard and 3. provide a consistent representation to facilitate communication among marketing, design, testing and development folks on the product team by providing a commonly understood foundation of concepts required to define, design and test the product from a user's point of view. Although we have used this technique successfully in several projects, reported evidence of the efficacy of this approach must await a future paper.

8 ACKNOWLEDGMENTS

The authors would like to thank Pete Lockhart for early contributions to the content of this paper and James Newbery for his extended reviews of earlier drafts of this paper.

9 REFERENCES

Holzenblatt, K & Jones, S. (1993) Contextual inquiry: A participatory technique for system design. In *Participatory Design, Principles and Practices*, D. Schuler and A. Namioka (Eds.) Lawrence Erlbaum Associates, New Jersey, USA.

Mitchell, C.M., & Miller, R.A. 1986, A discrete control model of operator function. A methodology for information display design, *IEEE Transactions on Systems, Man, and Cybernetics*, **16**, 342-357.

Moore, G. A. (1991) Crossing the Chasm, Harper Business, USA.

Muller, M. (1993) PICTIVE: Democratizing the dynamics of the design session. In *Participatory Design, Principles and Practices*, D. Schuler and A. Namioka (Eds.) Lawrence Erlbaum Associates, New Jersey, USA.

Nielsen, J. & Landauer, T.K. (1993) A mathematical model of the finding of usability problems. In *Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings*, S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. White (Eds), Addison Wesley, pp. 206-213.

Newbery, James (1995) Contextual design v. systematic creativity: A short critique. Internal Intel Report.

Salvador, A. C. & Sundstrom, G.A. (1993) Information organization, search and management advantages of a task-based HMI for telecommunication network monitoring and

troubleshooting. *Proceedings of the 1993 IEEE International Conference on Systems, Man and Cybernetics.*

Sundstrom, G.A. & Salvador, A. C. (in press) Integrating field work and system design. *IEEE Transactions on Systems, Man, and Cybernetics.*

Sundström, G. A. (in press) Towards models of tasks and task complexity in supervisory control applications. *Ergonomics.*

Sundström, G.A. (1991) Process tracing of decision making: an approach for analysis of human-machine interactions in dynamic environments. *International Journal of Man-Machine Studies*, **35**, 843-858.

Sundström, G.A., & Johannsen, G. (1989). Functional Information Search: A Framework for Knowledge Elicitation and Representation for Graphical Support Systems.. *Proceedings of the 2nd European Conference on Cognitive Science Approaches to Process Control*, Siena, Italy, October 24-27, pp.129-140.

10 APPENDIX A: BRIEF DESCRIPTION OF INTERVIEW TECHNIQUE

Most elicitation techniques (e.g., verbal protocols, observations, etc.) are sufficient to extract the information necessary for definition, design and usability testing as required by the current framework, depending on time, resources and predilection. Although use of a particular knowledge elicitation method is independent of the framework posed in this paper, some methods are far more efficient and effective than others due to the volume of information required. For example, many methods capture information that is not necessarily required for product definition, design or testing. The current framework requires a minimal, but very specific set of information, thus a minimal but very specific elicitation method should probably be employed. A brief description of the technique we use is described.

Typically, two interviewers conduct the interview, one leads questioning and one records responses according to the framework. The recorder and interviewer work closely together to ensure that missing cells are filled in and that all branches are explored. An interview lasts for about 1.5-2 hours. Interviews should be performed at the user's location in their work environment, within sight of the artifacts they use daily. At the beginning of the interview the interviewers describe the process to the user. During the interview, the interviewer asks some open-ended and some detailed questions. Throughout the interview, the user is to expect numerous interruptions that will save both parties time.

The first series of questions involves the general description of what the user does for a living, i.e., what their role in the organization is. During this time, the interviewers are listening, reorganizing the user's monologue into a series of user goals, objectives, tasks, facilitators and obstacles. The interviewer will ensure that the user describes the breadth of his/her work before following any particular thoughts in depth. At some point when the user looses steam, the interviewer begins to repeat the user goals back to the user, at which point the user invariably notices where the description is lacking and continues. This cycle continues until the user and the interviewer are satisfied that no more can be gained at this time.

Subsequently, the interviewers probe each user goal to elicit objectives, tasks, subtasks and facilitators. Each goal is probed in turn, using to the extent possible the artifacts in the work environment as stimulus for continued description. At all times, during the interview, the second interviewer is restructuring and recording the user's description to conform to the framework, which is then used by the interviewer to identify gaps in the user's description.

11 BIOGRAPHY

Tony Salvador co-manages the Human Factors Services group in Intel's Personal Conferencing Division. Dr. Salvador is currently responsible for user work practices requirements analysis, user interface design and usability testing for Intel's ProShare Personal Conferencing product line. Prior to this, he worked at GTE Laboratories, Inc. where he performed requirements analysis and user interface design for GTE's centralized network management system.

Jean Scholtz co-manages the Human Factors Services group in Intel's Personal Conferencing Division. She is currently on leave from Portland State University. At Portland State, she initiated a master's track in Human-Computer Interaction. Dr. Scholtz is responsible for user work practices requirements analysis, user interface design and usability testing for a range of multimedia products including Intel's CNN at Work. She is active in the CHI community, both locally and internationally. She is the co-program chair for the industry track of INTERACT '95.

Discussion

Hong-Mei Garcia: What kind of products are you talking about here?

Jean Scholtz: Any kind of product.

Hong-Mei Garcia: Is it organization dependent?

Jean Scholtz: No.

Hong-Mei Garcia: Do you screen users before you select them?

Jean Scholtz: We do.

Christopher Ahlberg: How far do these ideas generalize?

Jean Scholtz: Intuition is that it generalizes to most types of design.

Pedro Szekely: What kind of tools do you use to record the information?

Jean Scholtz: Word, spreadsheets. Tools are planned

Peter Jensen: How do you get users to do your work?

Jean Scholtz: We pay them.

Rick Kazman: Do you split goals into immediately achievable/Release 2? If so, are goals for future releases incorporated into the design?

Jean Scholtz: Goals are prioritized. Goals that do not make it into release one are left as hooks.

Len Bass: Where does your product concepts come from?

Jean Scholtz: Currently, most ideas come from engineering and goes through a review process that involves marketing and human factors.

Christian Gram: When you do user interviews are you hindered by corporate confidentiality since you cannot tell the users what new products you are thinking of?

Jean Scholtz: Yes, we have a lot of confidentiality and users sign a non-disclosure form.

Stephen North: Can you be more specific about how your methodology affected the products in which you used it, for example, ProShare?

Jean Scholtz: For example, we might learn whether users consider an interactive query facility to be important or not , whether they would use it every day. It is difficult to answer your question without revealing proprietary information.

Pedro Szekely: Can you say something about the size of database for ProShare (number of goals, tasks).

Jean Scholtz: No.

Pedro Szekely: What is the difference between customers and users?

Jean Scholtz: Customer writes the check, users work with the product.

Prasun Dewan: Do you have something in your framework for looking at a competitor's product?

Jean Scholtz: Yes.

Prasun Dewan: After the fact or before?

Jean Scholtz: At the same time.