

# Communicating Requirements Using End-User GUI Constructions with Argumentation

J. Michael Moore  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112 USA  
1 979 845 5534  
*{j michael@cs.tamu.edu}*

## 1. Introduction

Unsuccessful communication is often at the root of inadequate requirements specification [14]. This can lead to requirements that do not capture complete stakeholder expectations. Stakeholders can include managers, software engineers, end-users, clients, etc. End-users provide a rich source of information about a system as they will directly interact with the final system. They also tend to have a solid knowledge of the domain including the tasks being automated. Thus, a major goal early in the software engineering process is gathering meaningful requirements from end-users.

Potts, et al. [14] propose inquiry-based requirements analysis. They break the process into three major phases: Requirements Documentation, Requirements Discussion, and Requirements Evaluation. Finkelstein [5] talks about seven difficult areas in requirements engineering. One of those areas is requirements acquisition. Requirements documentation or requirements acquisition is the process of getting an initial set of requirements from stakeholders. Getting this starting base of requirements is a challenge.

My topic is to explore using mock end-user graphical interface construction supplemented with textual argumentation as a means of communicating software requirements information to software requirements analysts and providing automated assistance for requirements analysts examining this information.

## 2. Requirements Gathering

Zave and Jackson [21] discuss areas where requirements research is weak. They note that terminology used for requirements should match the environment where the machine (e.g. software) is going to be built and used. Language is often ambiguous, especially when peppered with jargon, i.e., technical

terms used within a group. End-users and requirements analysts essentially speak two different languages. These languages may share terms that are used in different ways by each. End-users and requirements analysts can become involved in a language game where each does not know the rules of the other [4]. Successful communication is hindered when textual language alone is used for communication.

Combining communication with reference to an artifact can facilitate communication by allowing "design by doing" [4]. Moreover, Glenberg and McDaniel [6] have noted that integrating spatial and linguistic information is a requirement for effective communication. Others have used a combination of visual and textual information. Reeves and Shipman [15] use visual design artifacts as the focus for communication. Shipman, et al. [17] use visual objects for information organization and interpretation.

Requirements state expectations at two levels. The first level consists of high-level functional requirements that state goals the system will achieve. The second includes more fine-grained procedural expectations that describe how the system will behave. It is possible to fulfill a set of functional requirements in multiple ways using different instantiations of procedural requirements or behaviors.

### 2.1. Other Approaches

Many techniques are available eliciting and refining requirements [12]. Methods for obtaining the initial set of requirements can include questionnaires, interviews, task analyses, and goal based methods that employ a rigid rule-based approach [3]. Each method results in a set of requirements that capture functional and procedural information to varying degrees.

While users know the functions that the software should include, they do not explicate the fine-grained

procedural behavior that they desire. Sometimes they assume that their high level functional descriptions entail the detailed procedural steps and processes that are not made explicitly. Moreover, there may be tacit knowledge that users cannot share [13]. Getting at this useful and necessary information is inherently difficult and compounded by the inexact nature of language. Much of the ambiguity in these descriptions can be resolved through further communication.

Communication, whether it is in writing or face-to-face, assumes a shared background knowledge supporting the exchange of information. However, when attempting to elicit detailed procedural information, these assumptions can inhibit the successful exchange of ideas. When attempting to elicit information from a client using existing requirements specification processes, the client may not verbalize pertinent information assuming that the receiver knows these things implicitly.

Face-to-face communication allows for the rapid repair of communication breakdowns [18]. However, these interactions may limit the exchange of some information due to the influence of the “expert” software engineer. The end-user may be led down a specific path that matches the expectations of the software engineer.

Task analysis can be used to obtain information about work practices, even practices that end-users are not aware of. Ethnographic analysis gives a requirements analyst access to the rich features of human communication mentioned above. Unfortunately, some of these benefits are lost since methods where the observer asks questions can cause people to break from the routine that the observer is trying to capture.

While interviews and task analyses are a rich source of requirements information, they can be time consuming

since they require users and software designers to be co-located in time and space. The related scheduling issues can draw out the requirements analysis process, lengthening the overall software development timetable.

The Requirements Apprentice (RA) develops a coherent internal representation of a requirement from an initial set of disorganized, imprecise statements [16]. The initial set of data input by the software engineer is based on interviews with end-users. RA does not interact directly with end-users. Although RA works to provide a better set of initial requirements, it still relies on existing methods of initial requirements acquisition, i.e. questionnaires or interviews.

Interviews and questionnaires provide needed information but place the user in the role of informant rather than participant [11]. Some software development processes have been shifting towards participatory design (PD), where users take a more active role [1]. Many PD activities have focused on design instantiation and more downstream activities [2][20]. These activities have included modifying prototypes [8][19] and paper designs of interfaces [11]. One major advantage of PD is that participants form a personal stake in the product and are more likely to work to make it succeed. It helps develop a sense of ownership within the user community. As with interviews and task analyses, participatory design requires user time. Thus, it can be difficult to schedule and costly.

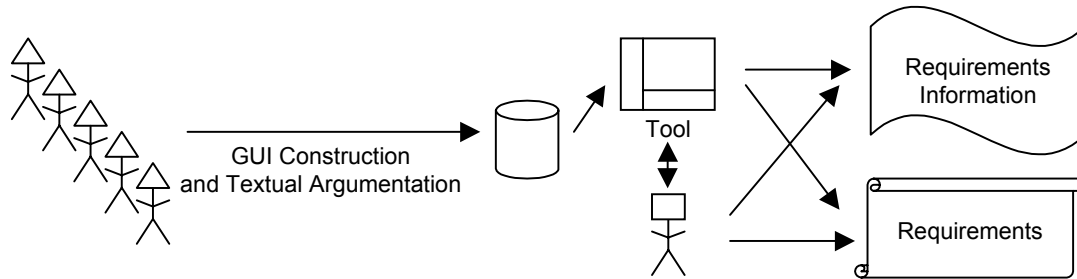
The problem is that requirements gathering methods tend to fall into two categories: those that produce rich results but are expensive (in time and money) and those that are less expensive but also less informative. A tool that allows initial requirements gathering to occur outside the realm of face-to-face interactions while generating a richer initial set of requirements has the potential to reduce the amount of user time actually needed.

### 3. Approach

My approach is to provide a software-based requirements acquisition tool that interfaces directly with end-users. The end-users will create mock user interface constructions augmented with textual argumentation that will act as communication to software requirements engineers (Figure 1). A second tool will provide software requirements engineers support for analyzing mock user interfaces constructed by multiple end users. Figure 2 shows an overview of this process.



Figure 1: End-user design with textual argumentation for “Finished With This Window” button



**Figure 2: Approach**

It is important that users feel involved and have a feeling of control during the requirements definition process [7]. In my approach they will feel more control over the requirements since they work with a computer and not with an "expert" who might influence them. Furthermore, this approach gives end-users an active role in the requirements gathering process, as with PD.

The use of graphical user interfaces (GUIs) has become ubiquitous through the use of operating system interfaces such as Microsoft Windows and the Macintosh OS. Through this interaction paradigm the usage of widgets has become standardized to the point where the placement and usage of widgets are essentially a visual language, the "language of the GUI." People who regularly use GUIs implicitly understand this language. Users see radio buttons and think "one choice" where software developers see "mutual exclusion." The language of the GUI allows communication through a more unified language with a vocabulary of widgets. Moreover, requirements analysts can identify how textual vocabulary, as presented by end-users, fits into their understanding of the environment. The language of the GUI facilitates the transfer of these concepts from end-user to software developer, thereby avoiding many of the trappings of jargon and textual communication.

By using the language of the GUI, interface construction becomes a way for end-users to communicate their desires and goals for software. End-users can convey things that might be too difficult using solely text. This visual approach may also be insufficient for adequate communication. It can be clarified by attaching textual argumentation to graphical artifacts, allowing the strengths of both textual and visual information to augment communication.

Some may argue that interface construction is a design task and not a requirements gathering activity. While end-users *are* basically designing interfaces, most end-users are *not* skilled or trained in interface design. Regardless of the design quality, they are creating interfaces that can be reverse engineered to identify salient requirements. The language of the GUI allows communication of these requirements through end-user designs. This should be

viewed as *design for elicitation* rather than the final interface design.

Within the context of the direct interaction between end-users and the system, end-users provide both graphical design ideas and textual argumentation that are analyzed to find relationships and requirements. This can create a large data set that can be daunting to analyze. Providing support for analysts without hindering creativity is a challenge.

First, requirements analysts need access to the various constructions created by users including the ability to browse various user constructions. Some of this navigation can be based on the evolution of the workspace. Analysts can look not only at the final design, but also look at how user constructions evolved over time. This can provide insight on what users were thinking when developing mock interfaces. Access to work done that was later removed through the undo mechanism might also provide insight. Moreover, search facilities are needed to rapidly move from place to place in the information space created by users' constructions. One way is to provide indexes into the space based on characteristics of user constructions. Characteristics include: argumentation, author, widget types, widget labels, spatial layout information, and temporal information. Analyzing user constructions serially could be quite laborious. Rather, analysts should be able to search based on the characteristics of constructions and see how different users presented that content.

As analysts work through the requirements data, they often create formal representations. For example, if using UML they may produce class hierarchies, activity diagrams, and collaboration diagrams. They may also produce use cases and scenarios. This process can be time consuming. Automated analysis of the workspace can provide preliminary formal representations that give a starting point for further development. Developing formal representations is tricky. It is unlikely that formal representations created automatically from loosely structured data will be accurate enough for use without refinement. However, automated creation may reduce the time necessary to develop formal representations.

## 4. Contribution & Current Results

A gap exists between requirements elicitation techniques. My approach will provide an intermediate option that falls somewhere between questionnaires and face-to-face techniques. The tools I create to implement this approach will provide support for requirements generation from multiple end-user constructions.

In addition to the contribution of a software system for requirements elicitation, my work will provide insight into the viability of this approach. Finally, it will provide insight into end-users' understanding and expression of software requirements as well as software analysts' comprehension of end-user expressions.

Some of this work has been reported in [9] and [10]. The elicitation tool has been developed and used to collect data from approximately 75 students in three conditions. The first group provided information by creating interfaces that could be augmented with textual argumentation. The second group was limited to using only text. The third group was limited to creating interfaces without textual argumentation.

Currently the analysis tool is under development. Current work is focusing on how to use textual, spatial, temporal information, as well as the semantics of the language of the GUI individually and in combination to ascertain requirements information. Some of this work is focused toward creating diagrams that might be of interest to a requirements engineer.

## 5. References

- [1] Carroll J.M., Rosson, M.B., Chin, G. and Koeneemann, J. (1997) Requirements Development: Stages of opportunity for collaboration needs discovery, *Designing Interactive Systems: Processes, Practices, Methods, & Techniques*, Proceedings of DIS'97, pp. 55-64.
- [2] Chin, G., Rosson and M.B., Carroll, J.M. (1997) Participatory Analysis: Shared Development of Requirements from Scenarios, *Human Factors in Computing Systems*, Proceedings of CHI 97, pp. 162-169.
- [3] Dardenne, A., Fickas, S., and van Lamsweerde, A. (1991), Goal-directed Concept Acquisition in Requirements Elicitation, Proceedings of 6th International Workshop on Software Specification and Design, pp. 14-21.
- [4] Ehn, P. (1988) Playing the Language-Games of Design and Use-on Skill and Participation, *ACM Conference on Supporting Group Work*, 1988, pp. 142-157.
- [5] Finkelstein, A. (1994) Requirements Engineering: a review and research agenda, Proceedings of 1st Asia-Pacific Software Engineering Conference, pp. 10-19.
- [6] Glenberg, A.M. and McDaniel, M.A. (1992) Mental models, pictures, and text: Integration of spatial and verbal information, *Memory and Cognition*, 20(5), Psychonomic Society, Inc., 1992, pp. 458-460.
- [7] Holtzblatt, K., Beyer, H. (1995) Requirements Gathering: The Human Factor, *Communications of the ACM*, Vol. 38, No. 5, May, pp. 30-32.
- [8] Kyng, M. (1995) Creating Context for Design. In Carroll, J.M. (Ed.), *Scenario-Based Design: envisioning Work and Technology in System Development*, J. Wiley, NY, pp. 85-107.
- [9] Moore, J.M. and Shipman, F.S. (2000) A Comparison of Questionnaire-Based and GUI-Based Requirements Gathering, *Proceedings ASE 2000, The Fifteenth IEEE International Conference on Automated Software Engineering*, 2000, pp. 35-43.
- [10] Moore, J.M. and Shipman, F.S. (2001) Requirements Elicitation using Visual and Textual Information, *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, 2001, pp. 308-309.
- [11] Muller, M.J., Wildman, D.M., and White, E.A. (1993) 'Equal Opportunity' PD Using PICTIVE. *Communications of the ACM*, Vol. 36, No. 4, pp. 64-66.
- [12] Nuseibeh, B. and Easterbrook, S. (2000) Requirements Engineering: A Roadmap, *ICSE-2000 The Future of Software Engineering*, A. Finkelstein (ed), Limerick, Ireland, 4-11th June 2000.
- [13] Polanyi, M. (1966). *The Tacit Dimension*. Garden City, NY: Doubleday.
- [14] Potts, C., Takahashi, K. and Anton, A.I. (1994) Inquiry-Based Requirements Analysis, *IEEE Software*, Vol. 11, Issue 2, pp. 21-32.
- [15] Reeves, B. and Shipman, F. (1992) Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces, *Proceedings CSCW '92, ACM 1992 Conference on Computer-Supported Cooperative Work*, Toronto, Canada, 1992, pp. 394-401.
- [16] Reubenstein, H.B. and Waters, R.C. (1991) The Requirements Apprentice: Automated Assistance for Requirements Acquisition, *IEEE Transactions on Software Engineering*, Vol. 17, No. 3, pp. 226-240.
- [17] Shipman, F.M., Hsieh, H., Airhart, R., Maloor, P., Moore, J.M., and Shah, D. (2001) Emergent Structure in Analytic Workspaces: Design and Use of the Visual Knowledge Builder. To appear in *Proceedings of Interact 2001*.
- [18] Suchman, L. (1987) *Plans and Situated Actions*, Cambridge, UK: Cambridge University Press.
- [19] Sutcliffe, A. (1995) Requirements Rationales: Integrating Approaches to Requirements Analysis, *Designing Interactive Systems: Processes, Practices, Methods, & Techniques*, Proceedings of DIS'95, pp. 33-42.
- [20] Wilson, S. and Johnson, P. (1995) Empowering users in a task-based approach to design, *Designing Interactive Systems: Processes, Practices, Methods, & Techniques*, Proceedings of DIS'97, pp. 25-31.
- [21] Zave, P. and Jackson, M. (1997) Four Dark Corners of Requirements Engineering, *ACM Transactions on Software Engineering and Methodology*, 6(1), 1997, pp. 1-30.