Research Methods
Project Literature Synthesis
Department of Computer Science
CSC4000W

Jared Cameron Baboo
BBXJAR001
Supervisor: Gary Marsden

May 8, 2009

Abstract

In designing interfaces that assist non programmers in programming, care must be taken to design a interface that is easy usable for the user. There exist several design methodologies which can be used to aid in achieving this. Looking at participatory and interaction design which have roots in ethnography we find that methods exist which aid the development of usable interfaces. From the point of an end user we also find that there studies that point to visual programming as a possible tool to facilitate the programming process.Relating this to the design of an interface to develop web tools for MXit we look at participatory design and interaction design and taking lessons from these and research in end user programming and can now begin to develop and efficient and user friendly tool for non programmers.

1 Introduction

In this report a summary of literature that relates to the design of interfaces for programming by non programmers is presented.More Specifically this report covers literature that details Design Methodologies and End User Programming and then shows the link that these have in relation to the problem domain with which this project is concerned.

1.1 Problem and Domain

MXit is a instant messaging client for mobile devices that has gained popularity and market share since it's inception. MXit has recently released their message API which allows developers the ability to write clients for the MXit service. This presents an interesting opportunity to take advantage of the popularity of mobile devices and MXit's large user base to provide some additional functionality and web services to the MXit community. However non developers who may benefit from MXit freeing up their protocol e.g. teachers, are at a disadvantage in using this functionality due to limited programming experience.

This project attempts to make this functionality available to non developers through the design of an interface that facilitates the development of a MXit bot which can offer services to MXit users. The problem domain that this paper covers includes the designing of an interface that allows non-developers to program services which can be applied to MXit with a bias towards the application of these services in the education field. This draws on and expands on the development of Dr Math.Dr Math is the name of the contact developed for the project named Math on MXit. Math on MXit is a m-learning tool accessible to learners in South Africa that assists learners where possible in mathematics. This takes advantage of the popularity of mobile devices and the low cost of GPRS communication[4, 3].

2 Related Work

2.1 End User Programming

End User Programming is programming done by someone who does not have programming as their primary job function. This is resulting as an increase in users who interact with computers daily having diverse requirements that result in them needing to program simple unique solutions with little to no experience[6, 20]. Research indicates that while this is undesirable it may be inevitable and indicates that there may be various steps that developers can take to make this an effortless experience for end users through improving the design of interfaces that the user would have to navigate in order to achieve their task[20, 18].

Research looked at focused on design aspects of improving end user programming. A common feature discovered introduced the notion of analogy, that is using analogy in programming to ease difficulties new or inexperienced programmers may have in programming[20]. What appears to augment this is the use of a visual programming language or visual interface which we look into and discover some opposing views. We discover that some people believe that Visual programming languages can be very useful and possibly a step forward in facilitating end user programming but that there are hindrances in the use and application of these languages[18].

Some research in this area indicates that the use of analogy, that is making something behave like another , can be useful in the design of end user programming systems. Research here indicates a move to make it easier for an end user to add features or scripts by extracting common features through stating a similarity relationship which they possess [20].

The implication of this is that it leads us to take an increased interest in understanding what the user knows and understands so as to use this to draw a bases for the analogy. While in some instances this can be easy e.g. using established ideas already introduced to the user. For new applications it implies extensive user studies to understand what the user knows which can be adapted to be used in the application.

Examining the practical values put forth in the literature could assist in the design of an interface as the ideas distilled can be used as a guide in designing interface targeted specifically at end users. This is may be useful since studying the users interaction and modeling the processes the user may go through in interacting with the system may produce many artefact's and the use of this principles may assist in breaking these down into manageable parts.This indicates that for this project as a viable solution to issues around end user programming Visual Programming languages may provide a possible solution or aide. Also this indicates that using things that the user is familiar with can be important in designing solutions to end user programming problems.

2.1.1 Visual Programming

As used in previously mentioned work a visual interface appears to be an interesting and viable method in resolving difficulties non programmers may find in programming as they can make programming easier and more intuitive [18, 23]. Visual programming makes use of a visual representation/annotation the programming process. Visual programming may be linked to the use of Programming by Example as seen in previously mentioned literature which indicate that while not a do all and fix all it may be the fire Prometheus stole from the gods to facilitate non programmer ease in programming [18].

An issue raised in the literature is that Visual Programming Languages may have little scope in use or re-usability for developing other programs as it is discovered that they may be problem domain specific and that the visualisation method chosen could not be suited for the problem that is trying to be solved [23]. Conversely it is also indicated that when a visual representation that well suits the problem is used it may improve user comprehension and efficiency. This does present a problem as it results in visual languages take significantly different design approaches and appearance for various areas of interest.

An advantage that visual programming languages may possess is that they present the user with a visual representations. Research indicates that this may be beneficial as programmers create visual mental models of problems and solutions which could be facilitated with the use of visual representations. A visual representation also implies a spatial representation as the spatial relationship between objects and their geographic locations may be clearly seen [19].

The implication is that when designing a visual language designers must take into account that it may not be a ideal or final solution. This is due to the fact that a visual representation may be well suited to one problem but not another. Artefacts used in a language could facilitate the solving of one problem or problem set but be inappropriate for another. However there may be evidence that using a visual programming language could be easier for novice programmers which shows the value of this in being used in relation to this project.

3 Design

3.1 Ethnography

In relation to design ethnography is a descriptive tool. It describes the way in which people behave. Ethnography is a technique developed in the social sciences to naturally observe settings and situations that is to observe situations with out disruption or intervention[11]. Ethnography can and has been successfully used to study various situations which shows that it may be a viable means of requirements elicitation[11], though ethnographers are wary in translating their observations into terms useful for design or the language used by ethnographers can be ineffective in assisting designers[1].

3.2 Participatory Design

Participatory design is a design process that attempts to enhance interaction between stakeholders(users) and developers. Participatory design encourages user participation in the design process making it a cooperative design technique. The benefit of this is that it improves knowledge acquisition since users are experts in work practices and empowers users as they will ultimately be the users of the system[16]. This design methodology aims to be as democratic as possible which could give rise to some issues. Another issue become the participants who take part in this process. It may not be possible for all people effected to take part in the process bringing in issues on how and why users participate in the process. Further issues may arise through the tools and methods used in analysis and design in this method[13]. Though there are tools and methodologies that can circumvent this. Examples of tools and methodologies that can be used are PICTIVE and the MUST method[15, 14]. PICTIVE is a technique that uses a combination of low tech objects and high tech recording tools. PICTIVE allows participants to design mock ups of the system that can be modified in real time[15].The MUST method is aimed at the design process with roots in ethnographic methodologies and which defines six principles that may be useful in design[14].

Participatory design can be useful in the design process. It increases user awareness and gives them part ownership of the project. Using the PICTIVE methodology can be especially useful as it allows the user the system is designed for an active say in the interface they will be using.In relation to the development of an interface non technical individuals can use it means that they can have an active hand in designing the interface which could improve usability of the interface for this group. As the project is focusing on education professionals having them participate in the design process increases our knowledge base of this group so as to better incorporate artefacts and metaphors which they can understand.

3.3 Interaction Design

Interaction design is a design process that focuses on people and their interaction with computer based systems.It studies peoples interactions with objects and systems. Through this it attempts to facilitate the design of more usable interfaces for users. This makes it a design methodology that focuses on understanding peoples interactions with systems and aims to improve the quality of their experience[9, 5].In the Jam-O-Drum Interactive Music System study[2] interaction design is shown to be an effective way in improving a design for an interface by observing the interactions between the system and the users. This study also highlights the notion that interactions can compete with each other for user resources as well as the importance of supporting new interactions through the provision of resources important to it's execution. This also shows that Interaction design is a requirements elicitation technique that can develop alternative designs to meet these needs[5]. There are two issues that may be important to take note of and that is the interactivity fallacy where the user is a component of the interactive systems that is being designed and the empirical fallacy where use is an activity open to empirical investigation[9]. This means that design objectives are not defined by user requirements and viewing the user as a component of a system blurs the line between functionality and usability.A tool that could facilitate the design of an interface is Lean Cuisine+[22].Lean Cuisine+ makes use of a semi formal notation that makes use of a dialogue tree to represent the interface. It is useful as it can both represent the interface as well as system responses.

Using interaction design can be useful in improving the design of an interface. What interaction design tells us is how users actually interact with systems which could be different than that envisioned by the designer. This can also be used to highlight issues that users may have in the interface with a aim of showing improvements that could be made. In regard to this project studying the users interaction with technology allows us to understand the level of understanding a user may have of existing technology with a aim to build on this understanding an interface that minimises user discomfort with new applications. Also the use of interaction design to refine an existing interface shows the advantages of using this.

An overview of these methodologies indicate that each has it's benefits and it's weaknesses. A commonality that Interaction design and Participatory design posses are roots in ethnography and the social sciences. Important for this project is designing an interface that works for the user and augments their roles. Participatory design has initial benefits in allowing the user take an active role in the design process and gives them part ownership in the project as well as increasing the knowledge base in this process. Interaction design is beneficial in understanding the way users interact with the system which can assist the designer in developing a more usable interface but this can only happen once an interface has been created or if the user uses a similar tool or technology. Taking this and looking at the research on end user programming we can see that applying the participatory design method to a small group first and then using interaction design to refine the design can result in a better interface. First the Participatory method is applied to gain a user buy in for the project as well as increasing the knowledge base for design allows the designer to implement an interface that is congruent with what the user already understands minimising the disruption it could cause. Following this with interaction design allows for the designer to refine and refactor the design so to improve the user experience.

4 Conclusion

In conclusion to designing a tool that aids non programmers in programming is not without it's difficulties. It is also not an impossible task. This paper demonstrates that there are methodologies in design that can facilitate the development of an interface though issues inherent in end user programming may persist. The methods described are based on ethnographic methodologies and while not without fault can be effective in design. Presented here was the idea of a visual programming language to assist user comprehension and ease as well as the use of analogy to facilitate this. In the development of a interface to facilitate non user programming the design methodologies can be sufficiently effective in assisting the designer in designing effective interfaces.

see also [7, 8, 12, 10, 17, 21].

References

[1] R. J. Anderson. Representations and requirements: The value of ethnography in system design. *Human-Computer Interaction*, 9(2):151, 1994.

[2] Tina Blaine and Tim Perkis. The jam-o-drum interactive music system: a study in interaction design. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 165–173, New York, NY, USA, 2000. ACM.

[3] L. Butgereit. Math on MXit: the medium is the message. *Plenary papers*, page 107.

[4] L. Butgereit. Using Instant Messaging over GPRS to help with school work.

[5] Enrico Coiera. Interaction design theory. *International journal of medical informatics*, 69(2-3):205–222, 2003. note: Working Conference on Health Information Systems.

[6] Michael Eisenberg and Gerhard Fischer. Programmable design environments: integrating end-user programming with domain-oriented assistance. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 431–437, New York, NY, USA, 1994. ACM.

[7] Howie Goodell, Sarah Kuhn, David Maulsby, and Carol Traynor. End user programming/informal programming. *SIGCHI Bull.*, 31(4):17–21, 1999.

[8] T. R. G. Green and M. Petre. Usability analysis of visual programming environments: A [']cognitive dimensions' framework. *Journal of Visual Languages & Computing*, 7(2):131–174, 1996.

[9] Lars Hallnos and Johan Redstrom. *Interaction Design: Foundations, Experiments.* Textile Research Centre, Swedish School of Textiles, Unversity College of Boros and Interactive Institute, Boros, Sweden, 2006.

[10] John Hughes, Val King, Tom Rodden, and Hans Andersen. Moving out from the control room: ethnography in system design. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 429–439, New York, NY, USA, 1994. ACM.

[11] John Hughes, Val King, Tom Rodden, and Hans Andersen. The role of ethnography in interactive systems design. *interactions*, 2(2):56–65, 1995.

[12] John A. Hughes, David Randall, and Dan Shapiro. Faltering from ethnography to design. In *CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 115–122, New York, NY, USA, 1992. ACM.

[13] Finn Kensing and Jeanette Blomberg. Participatory design: Issues and concerns. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 7(3/4):167–185, 1998.

[14] Finn Kensing, Jesper Simonsen, and Keld Bodker. Must: A method for participatory design. *Human-Computer Interaction*, 13(2):167, 1998.

[15] Michael J. Muller. Pictive—an exploration in participatory design. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 225–231, New York, NY, USA, 1991. ACM.

[16] Michael J. Muller and Sarah Kuhn. Participatory design. *Commun.ACM*, 36(6):24–28, 1993.

[17] Brad A. Myers. Creating user interfaces using programming by example, visual programming, and constraints. *ACM Trans.Program.Lang.Syst.*, 12(2):143–177, 1990.

[18] Brad A. Myers, Andrew J. Ko, and Margaret M. Burnett. Invited research overview: end-user programming. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 75–80, New York, NY, USA, 2006. ACM.

[19] MARIAN PETRE and ALAN F. BLACKWELL. Mental imagery in program design and visual programming. *International Journal of Human-Computer Studies*, 51(1):7–30, 1999.

[20] Alexander Repenning and Corrina Perrone. Programming by example: programming by analogous examples. *Commun.ACM*, 43(3):90–97, 2000.

[21] Martin Røscheisen and Christian Mogensen. Interaction design for shared world-wide web annotations. In *CHI '95: Conference companion on Human factors in computing systems*, pages 328–329, New York, NY, USA, 1995. ACM.

[22] C. J. Scogings and C. H. E. Phillips. Linking tasks, dialogue and gui design: a method involving uml and lean cuisine+. *Interacting with Computers*, 14(1):69–86, 2001.

[23] K. N. WHITLEY. Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages & Computing*, 8(1):109–142, 1997.