

Accessibility and Computing

A regular publication of the ACM Special Interest Group on Accessible Computing

Inside This Issue

- 1** A Note from the Editor
- 2** SIGACCESS Officers and Information
- 3** MyLexics: An Assistive Courseware for Dyslexic Children to Learn Basic Malay Language
- 10** Designing Smart Home Interfaces for the Elderly
- 17** Development framework for pervasive computing applications

A Note from the Editors

This issue contains three articles. The first paper by Muhammad Haziq Lim Abdullah et al., *MyLexics: An Assistive Courseware for Dyslexic Children to Learn Basic Malay Language*, describes an e-learning product that takes into account special needs of dyslexic children learning Malay language.

The second and the third contribution are closely related: Article by Callejas & López-Cózar, *Designing Smart Home Interfaces for the Elderly*, focuses on design of the smart home environments for the elderly people, while the paper by Václav Slováček et al., *Development Framework for Pervasive Computing Applications*, presents a technology that enables the implementation of such environments.

Adam J Sporka, Guest editor
University of Trento

Sri Kurniawan, Newsletter editor
University of California, Santa Cruz



SIGACCESS Officers and Information

Chair

Andrew Sears
Constellation Professor of Information
Technology and Engineering
404G ITE Building
Information Systems Department
UMBC
1000 Hilltop Circle
Baltimore, MD 21250, USA
Phone: +1-410-455-3883
Fax: +1-410-455-1531
chair_SIGACCESS@acm.org

Vice-chair

Enrico Pontelli
Department of Computer Science
New Mexico State University
Box 30001, MSC CS
Las Cruces, NM 88003, USA
Phone: +1-575-646-6239
Fax: +1-575-646-1002
vc_SIGACCESS@acm.org

Secretary/Treasurer

Shari Trewin
Accessibility and Software Productivity
Research
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
Phone: +1-914-784-7616
treasurer_SIGACCESS@acm.org

Newsletter Editor

Sri Kurniawan
Dept. of Computer Engineering
University of California Santa Cruz
SOE-3, 1156 High Street
Santa Cruz CA 95064, USA
Phone: +1-831-459-1037
editors_SIGACCESS@acm.org

Who we are

SIGACCESS is a special interest group of ACM. The SIGACCESS Newsletter is a regular online publication of SIGACCESS. We encourage a wide variety of contributions, such as: letters to the editor, technical papers, short reports, reviews of papers of products, abstracts, book reviews, conference reports and/or announcements, interesting web page URLs, local activity reports, etc. Actually, we solicit almost anything of interest to our readers.

Material may be reproduced from the Newsletter for non-commercial use with credit to the author and SIGACCESS. Deadlines will be announced through relevant mailing lists one month before publication dates.

We encourage submissions as word-processor files, text files, or e-mail. Postscript or PDF files may be used if layout is important. Ask the editor if in doubt.

Finally, you may publish your work here before submitting it elsewhere. We are a very informal forum for sharing ideas with others who have common interests.

Anyone interested in editing a special issue on an appropriate topic should contact the editor.

MyLexics: An Assistive Courseware for Dyslexic Children to Learn Basic Malay Language

Muhammad Haziq Lim Abdullah, Syariffanor Hisham, Shahril Parumo

Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka (UTeM)

{haziq, syariffanor, shahrilparumo}@utem.edu.my

Introduction

Dyslexia is a language-based learning disability resulting in people experiencing difficulties in reading, spelling, writing and speaking due to inability to differentiate sound components. These problems are sometimes accompanied by short-term memory difficulties, a lack of organizational skills and time management issues which all have an impact on learning. Dyslexia can occur regardless of age groups and in some cases; dyslexics may have higher IQ level compare to others.

MyLexics is a courseware that was designed to help dyslexic children learn to read and write in the Malay language. This project is collaboration between the Faculty of Information & Communication Technology, UTeM and Malaysia Dyslexia Association. The President of Dyslexia Association of Malaysia, Mrs. Sariah Amirin (1998 – present) is also the subject matter expert in the development of MyLexics's contents.

MyLexics was developed based the 'Dual coding theory' by Allan Paivio (Paivo and Begg, 1981) who suggested that a recall or recognition can be enhanced by presenting information in both visual and verbal form, combined with the Scaffolding teaching strategy (Wood et. al, 1976). – providing assistance to student on a as-needed basis, fading it as the competence increases. The courseware content has been structured as building-up process: The children learn the individual 'alphabets' and then combine the alphabets to make 'syllables', finally they add the combined syllables to other syllables to form 'words'. The implementation of stated principles via multimedia elements allows independent and interactive learning, and yet engages the learners in interesting tasks.

MyLexics is expected to contribute to the development of teaching technology in Malay language education for dyslexic children in Malaysia. This is aligned with the project by Ministry of Education Malaysia (MOE) called Dyslexia Pilot Projects (DPP) that was launched in March 2004 and currently being implemented in 30 government aided primary schools throughout the country. Apart from DPP by MOE, MyLexics is also in line with teaching activities conducted by non-government organizations, namely by the Kuala Lumpur Dyslexia Association. It has undergone an acceptance testing. The feedback and positive results from the dyslexic children at the center were very much promising.

Why MyLexics?

In Malaysia, it is estimated that 1 in every 20 students is dyslexic which means that each classroom in every primary school has at least 1 or 2 potential dyslexics (Kuala Lumpur Dyslexia Association, 2006). This is supported by the figures from the Ministry of Education that there are approximately 315,000 primary school children in Malaysia are potentially dyslexics. The courseware is not only benefits dyslexic children but also children with other

types of learning difficulties such as autism and slow learner. Table 1 shows the comparison between MyLexics and other similar products in the market.

Table 1. Product Comparison

MyLexics	Other Products on the Market
The interactivity and non-linear technique adopted in this courseware offers flexibility for parents and teachers in monitoring the children learning progress.	The linear-technique was adopted in most courseware and not suitable and inflexible for learning progress.
The non linear structure of the contents allows flexible navigation. Unlike many other similar products on the market, this interactive CD promotes self-learning which is suitable for home and school.	Most CD promotes guided-learning which not convenient to parents and teachers where they have to monitor the children learning progress.
Help using a video approach.	Limited of help in a CD.
MyLexics is suitable for main stream children and those having learning difficulties such as dyslexia, autism, ADHD and slow learner.	Only suitable for those not having learning difficulties children.

MyLexics: Courseware Modules

MyLexics is the first courseware in Malaysia for dyslexic children learning to read and write in Malay language. The courseware integrates all multimedia elements that supports interactive and self-learning environment both at school and at home. MyLexics comprises of *Modul Pembelajaran* which consists of three sub-modules (*Abjad*, *Sukukata* and *Perkataan*) are described in the further text.

The typical learning process is supported by the *Modul Pembelajaran*: It begins by learning the individual letters of the *Abjad* (Alphabet), consisting of vowels and consonants. Here, the important connection between the sound and the letter is made. Next, the children learn to combine the alphabets to construct two types of *Sukukata* (Syllables). Finally, syllables are used to construct *Perkataan* (Word).



Figure 1: MyLexics Main Menu

Abjad (Alphabet) Module

This module will give the users the foundation of recognizing and writing letters of the alphabet. First, they will be introduced to a letter and the shape of the letter. Then they will learn to recognize all the letters by its categories: Vowels and consonants. Among vowel letters, dyslexics always get confused with vowel 'e' and 'i'. This is due to the similar sounds of the letter (Mohd Yusoh et. al, 2008). Thus, the letters are displayed in separated pages. Figure 3 shows a sample screen for writing a vowel.

The users can click on the letters on right hand side to start a 2D animation of letter writing. A voice over will pronounce the letter and dashes lines of the letter will be provided on screen as guidance. Dyslexics can place their fingers on the screen and follow the 2D animation to learn writing the letter. Here, the visual and kinesthetic elements used will reinforce each other for optimal learning. This involves a creative, participatory act by the dyslexics.

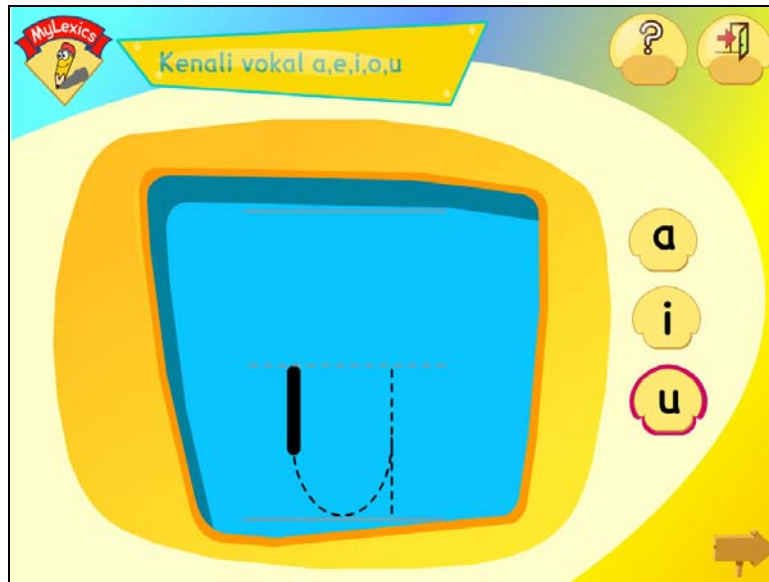


Figure 3: Introducing and Writing a Vowel

Dyslexics always rely on pictures and contextual clues to say a word (Earnshaw and Seargeant, 2005). This is due to their poor decoding skill especially of symbols like letters. To overcome this problem, MyLexics associated letters with images and audio. Here, dyslexics will associate the shape of the image with the letter itself. This is an example of dual coding theory concept. The dyslexics are given cues by images and audio to help them recognize letter that they do not immediately recognize. Figure 4 shows a sample screen of recognizing a letter in Consonant Module.

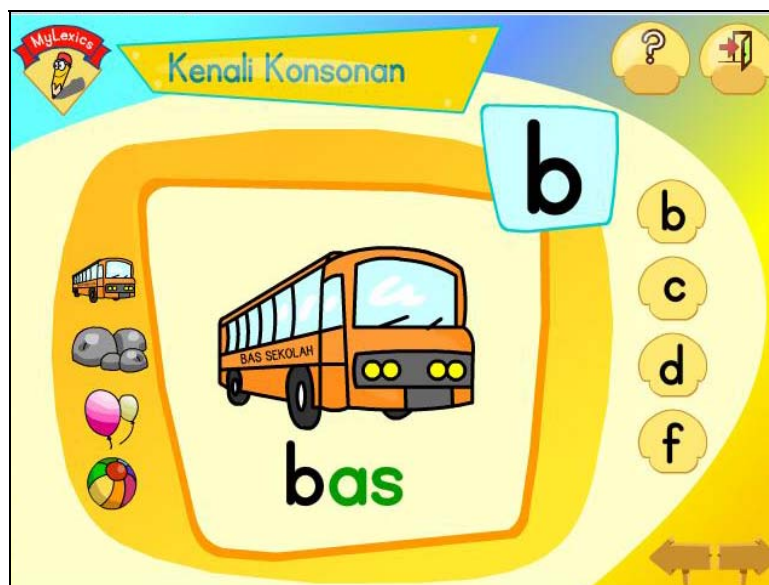


Figure 4. Recognizing a Consonant

Sukukata (Syllables) Module

In *Abjad* Module, dyslexics learn the individual vowel and consonant and its sound. In this module, they will learn the combination of a vowel and a consonant to form a syllable. As MyLexics is targeted to preschoolers, only two-letter syllables are covered. The method used is quite straightforward. In MyLexics, dyslexics will learn syllables using Simultaneous Oral Spelling (SOS) technique. This technique stresses four main components of learning, which are 1) Hear it, 2) Say it, 3) See it, 4) Write it. Here, the SOS uses multi-sensory approach where learners use their vision and hearing to learn the syllables. In addition, in order to capture and sustain dyslexics' attention and interest, simple animations are provided with interesting illustration of the letter combination process. Figure 5 below shows sample screens for *Sukukata* Module.

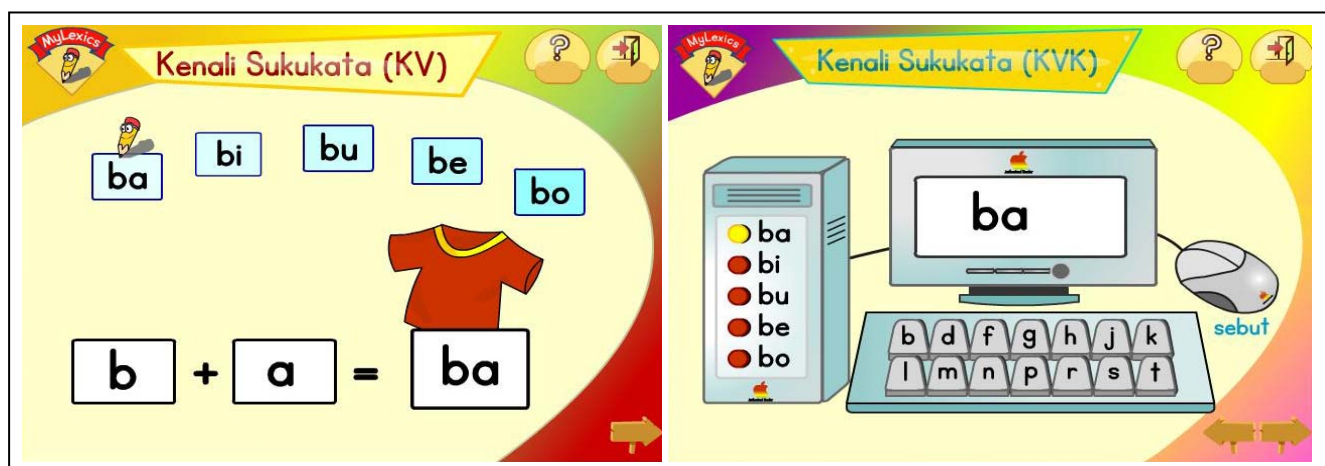


Figure 5. 'Sukukata' Module

The learner can click on the syllable, hear the sound of syllable, and watch a 2D animation of combination of the letters to make the syllable. Like in *Abjad* module, vowel 'i' and 'e' are displayed in separate screen.

Perkataan (Words) Module

This module will teach the dyslexics how to read simple words. As of this stage, we expect that the learner has acquired the skills of recognizing letters and basic syllables. The teaching approach in this module is by using family group of words (e.g. suka, buka, cuka, luka). Family group of words are words that have syllables that sound very similar. The aim is that the dyslexics will be able to recognize the pattern of family word instantly. Like other modules, *Perkataan* module also uses images, text and audio. First, a set of words and a picture associated with the word is displayed. Here, the dyslexics will first figure out the story content from the pictures. This is an example of exploratory driven based learning (Mohd Yusoh et. al, 2008). The learner will make the transition to learn the syllables that make up the word. Each word contains two syllables represented in two standard colors used in the Dyslexia Association Kuala Lumpur, Malaysia. Once the learner understands the material, words become more meaningful. This is an example of inquiry based learning

which will help the dyslexics to retain more information. Next, the system will highlight the next picture with an animation of letter transition in the first word. Voice-over is provided to explain the transition. Figure 6 shows the word set. The learners can continue with the next family word by clicking at the arrow button.



Figure 6. Perkataan Module

Comparison with other products

There are several products on the Malaysian market to teach reading such as *Siri Cepat Membaca Bacalah Anakku* and ReadEasy (<http://readnetwork.com>). *Bacalah Anakku* is a package of basic learning of Malay language for children, consisting of 8 books, 1 booklet, 1 VCD and 12 scan cards. The comparison between MyLexics and *Bacalah Anakku* are shown in the Table 1.

Recognition of the initiative

- Silver Medal, E-learning for Disability: An Assistive Multimedia Courseware for Dyslexics, UTeM Exhibition 2008 (In conjunction with Malacca Education Carnival), Innovation Category, Melaka International Trade Centre, 27-30 March 2008.
- Gold Medal, MyLexics: An Assistive Multimedia Courseware for Teaching and Reinforcing Basic Skills in Reading Malay Language among Dyslexics, The 18th International Invention, Innovation and Technology Exhibition (ITEX07), Kuala Lumpur Convention Centre, 18-20 May 2007.
- Silver Medal, E-learning for Disability: An Assistive Multimedia Courseware for Dyslexics, Teaching Methods and Materials Category, The 35th International Exhibition of Inventions, New Techniques and Products of Geneva, 18- 22 April 2007.

Acknowledgement

This work is funded by UTeM under the Short Research Grant Scheme by Centre for Research and Innovation Management (CRIM), UTeM.

References

1. Paivio, A. & Begg, I. (1981). *The Psychology of Language*. New York: Prentice-Hall.
2. Wood, D., Bruner, J., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17, 89-100.
3. Mohd Yusoh, Z.I., Devaraju, A., Zakaria, M.H. & Techanamurthy, U. (2008). *An Overview of Learning Contents in MyLexics - An Assistive Multimedia Courseware for Dyslexics*. In: Proc. The 13th International Conference in Education, Brunei Darul Salam
4. Earnshaw, T. & Seargeant, A. (2005). *Dealing with dyslexia and other reading difficulties*. Prentice Hall
5. ReadNetwork, <http://readnetwork.com>

About the Authors

This article was written by Muhammad Haziq Lim Abdullah, Syariffanor Hisham, Shahril Parumo. In addition to the authors of this article, other researchers who were involved in this project are Wan Sazli Nasaruddin Saifuddin, Mohd Hafiz Zakaria, Anusuriya Devaraju and Zeratul Izzah Mohd Yusoh.



Muhammad Haziq Lim Abdullah is a lecturer and researcher at the Faculty of Information and Communication Technology at Universiti Teknikal Malaysia Melaka (UTeM). He has a Master of Multimedia from Swinburne University of Technology, Australia and Bachelor of Information Technology (Honours) (Multimedia Studies) from Universiti Kebangsaan Malaysia. His research interests are in the area of assistive technology and artificial intelligence.



Syariffanor Hisham is a lecturer and researcher at the Faculty of Information and Communication Technology at Universiti Teknikal Malaysia Melaka, Malaysia. She has a Bachelor Degree in Information Technology (Universiti Utara Malaysia) and an MSc in Multimedia Technology from the University of Manchester. Her PhD thesis at the University of York, UK is about Localising User-interface design for elderly users in Malaysia. Her research interests are mainly in the area of Human-Computer Interaction including assistive technology, culture and people with special needs.



Shahril Parumo is a lecturer and researcher at the Faculty of Information and Communication Technology at Universiti Teknikal Malaysia Melaka, Malaysia. He has a Bachelor Degree in Computer Science (Universiti Teknologi Malaysia) and a MSc in Computer Science from the Universiti Putra Malaysia. His research interests focus towards applications of information technology in health and human development.

Designing Smart Home Interfaces for the Elderly

Zoraida Callejas, Ramón López-Cózar

Dep. Languages and Computer Systems
University of Granada (Spain)

{zoraida,rlopezc}@ugr.es

Abstract

In this paper we highlight the importance of tailoring the design of dialogue systems to the targeted user group. We propose a human-centered design cycle and report the results from a survey conducted among the intended users of a smart home for the elderly.

Introduction

The use of ubiquitous computing at home allows the automation of routine activities, removing physical barriers and adding benefits provided by mechanical and electrical technologies (Aghajan et al, 2009). This is very useful for the elderly, as the living in a smart home can ease their everyday activities and in some cases might be the only way of enjoying autonomy without risks.

To ease the interaction for these users, we are intending to set the basis for the design and development of a ubiquitous multimodal dialogue system to assist them in their daily home activities. Communication with computers using natural language eases the interaction notably and is a key issue in the efforts towards the so-called e-integration and e-accessibility initiatives. By means of these new technologies everybody will be able to interact with smart homes, especially people who have problems using these new technologies due to ignorance, disregard or disabilities (López-Cózar and Callejas, 2010).

Our proposal to design for usability

One key aspect for the development of interfaces is putting the emphasis on usability. Usability clearly differs from other system quality measures such as maintainability, reliability or portability in that the user point of view plays a determinant role in it. As a result, the objective of the interactive development cycle must not be only to obtain fully functional systems, but must also be concerned with the development of systems which adapt to the user needs, expectations and goals.

Thus, classic machine-centred approach, in which the system design and implementation are carried out following the technological aspects and functional user requirements, is no longer suitable. It becomes necessary to incorporate issues concerned with user capabilities, capacities, context and preferences which constitute the basis of human-centred design.

Several studies (Shaffo and Brown, 2002) have shown that the life-cycle cost of complex systems is affected in approximately 70% by the decisions made in the early design stages. Thus, to mitigate life-cycle costs and risks it is very important to create user performance models that let the system designers quantify the real operation scenarios at the early design stages.

Our proposal considers three key features of a system: interaction management, user modelling and multimodal interface. On the one hand, the interaction management establishes a relationship between system and context. Thus, e.g. to interpret a user gesture the system must have information about the dialogue and the social and situational contexts. On the other hand, the user models establish a relationship between users and their context. Thus, if the user is an elderly person, it is important to know e.g. whether s/he leaves alone and his/her disabilities. Finally, the multimodal interface provides a way for system-user communication.

We propose to carry out the tasks shown in Figure 1, which can be described as follows. In the first step, the system designers arrange a preliminary set of requirements based on their expertise on multimodal dialogue systems. In other words, they specify necessary and desirable functionalities and properties for the system. The second step is to refine these requirements taking into account the final users' points of view about the proposed functionality and characteristics, including also their new ideas and suggestions. The refined set of requirements allows in a third stage the creation of a good system specification. The fourth step is the construction of a first design of the system that complies with the specification. Finally, the fifth step uses the so-called Wizard of Oz (WOZ) technique (Dow et al, 2005) to find the usage context of the preliminary design, which is incorporated into posterior designs in an iterative way.

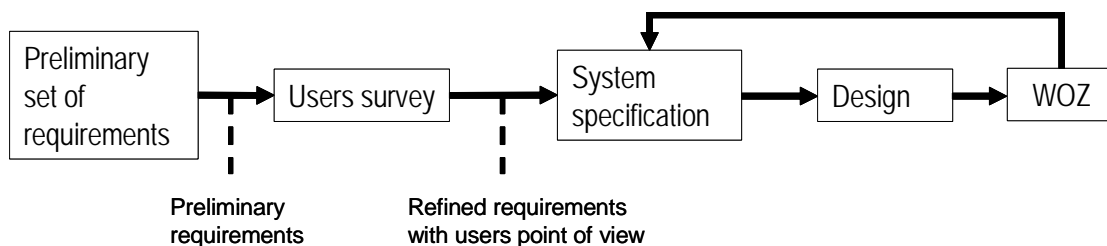


Figure 1. Tasks to implement the proposed approach to system design.

A case study: design of an interface of a smart home for the elderly

In the preliminary requirement analysis of our system we established the functional requirements to include the multimodal control of home appliances such as lamps, ovens, windows and heating. We also considered the system should include some entertainment functionality, in our case music and TV remote control and programming. We also found interesting that our multimodal system included a directory for helping the elderly users to make telephone calls. This way they could utter the name of the person they wish to call without remembering his/her telephone number.

Once we had restricted the problem, we reduced the possible solutions by establishing a set of non-functional requirements. Therefore, a suitable system that accomplished all the functional restrictions should also have the following characteristics: *non intrusive*, i.e. it would not interrupt users in their activities unless necessary, *proactive*, i.e. it would decide when to help users autonomously considering the environment's current status, *adaptive*, i.e. it would take into account the environment conditions in terms of noise and lighting to decide the most reliable interaction modalities. The system would also allow external communication, e.g., it could make emergency phone calls, and would accept incoming

user calls to operate several devices (e.g. switching the heating on before the user arrives home). Finally, to enhance the user experience, it should be *customizable*, *friendly* and *easy to use*.

Users survey

To refine the preliminary requirement as shown in Figure 1, we carried a user survey among 200 men and women with ages ranging 50-80 who lived in towns of different sizes. Most interviewed were aged 60-70, 58% women and 42% men. The survey includes a representative sample of different segments of age, studies, economic situation and residence place, to obtain results that could be generalized for all the intended users of the system.

The survey was created following several style rules. The first was the usage of an easy language so that the questions it contained could be perfectly understood by every polled, potential user. Thus, we avoided foreign or very specialized words. As the survey was directed to elderly, we avoided asking about something that had been explained long ago, and employed very concise questions. Besides, questions were ordered from general to specific, clearly divided in different thematic areas. Due to the difficulty for the elderly to understand concepts related to new technologies, we explained every idea and illustrated it with a drawing representing a person interacting with the system.

We carried out the survey in two ways: one was by giving a printed copy of the survey to the polled, while the other was interviewing them orally. For the former we used large fonts for the text so that it was easy to read for them. For the latter, an interviewer read the form and took note of the polled answers and opinions.

Firstly, a section introduced the system we are planning to develop and explained why the user collaboration was very important for the design. Secondly, there was a section with questions related to personal details: age, gender, residence place, studies, labour situation, among others.

A third section contained questions about the utility of the different system functionalities were enclosed in seven categories: illumination, temperature, windows and blinds, music, television, kitchen and contact. Illumination referred to the ability of the system to control illumination via multimodal commands (e.g. switch the lights on and off), temperature referred to remote control of the house temperature, windows and blinds referred to remotely opening/closing them without the user having to make any effort. Music and TV were the system functions that enable the user to control and program the radio and TV sets (e.g. choose TV channels or radio stations). In the kitchen category we enclosed the control of the main home appliance such as ovens, washing machines and fridges. Finally, in the contact category we include the system's ability to act as a telephone agenda. For each of these categories, the survey started with a brief paragraph describing the system function followed by a drawing which explained the concept graphically. Then the polled people were asked if they found this function useful and how frequently they would use it. They were also asked to justify negative answers. Furthermore, a blank space allowed them to indicate other possible functionalities associated with the category.

The last part of the survey considered system characteristics rather than functionalities. The polled were asked to indicate if they found useful: i) the system's exterior access (e.g. to order switching the heating on before s/he arrives home), ii) proactiveness (the system

ability to carry out tasks without being asked to, e.g. to remind things to the user by its own initiative), iii) human appearance (using an animated agent to generate a visual system output), iv) ease of use, v) customizability, vi) user location (system ability to know where the user is at every moment, e.g. to decide if s/he forgot to switch off a light) and vii) recovery from interaction errors. Finally the polled were asked to include comments or suggestions about the system functionality.

The survey results are shown in Figure 2 and 3, which illustrate the perceived usefulness of the main systems functionalities and their intended frequency of use respectively.

The figures show the results are very encouraging, as in all the cases the polled people found the functionalities useful, although their usage varies (e.g. music control was found useful but it would not be used very frequently). The telephone agenda was the most useful utility in the potential users' opinion, even for who indicated not suffering from memory problems. A reason for that might be that this kind of application is better known, so the polled distinguished with more confidence that it was really useful for them.

It was specially unexpected for us the acceptance of the windows and blinds function by the polled people, which was explained to them as the ability of the system to open and close windows and blinds automatically by means of voice commands. We think the reason for this result is that it is a very arduous task for the elderly. This is especially true for those who suffer from motor disabilities (124 out of 200 polled). Another very well accepted function is the TV control using spoken commands. We think the reason is that watching TV is the most widespread activity between the elderly (more than 90% of the Spanish older than 65 watch TV every day).

However, the ability of the system to control kitchen appliances (e.g. ovens) was not very well accepted as most people answered they would never use this function. We think the reason is that most of the elderly do not make the home duties alone (as they indicated in the first part of the survey), either because they share them with other family members, or because somebody does that work for them. We found that 50% of people who make all duties (usually women) considered this function really useful, while 69.4% of the polled who do not do the activities at home said they did not find it useful and would never use the system for those tasks.

We also found that, in general, all the system characteristics (non functional requirements) were broadly accepted: access from the house exterior (52%), reminder of things to users (91%), user location within the house (59%), personalization (58%), emergency calls (84%). These results show the most broadly accepted system characteristic is its proactiveness. On the contrary, the animated agent was not well accepted (54% of acceptance, 37% non acceptance) as many of the interviewed people remarked they preferred not to have a human image of the system when they interact with it. Furthermore, 95% of interviewed said they would consider the system easy to use because of its multimodal interface.

The potential users were also asked what they would do if the system misunderstands them. Surprisingly, most said they would repeat the same thing until it finally understood. The top second answer was to shut the system down. This fact clearly shows the difficulties in developing a multimodal dialogue system for the elderly, as they would not trust the system if it fails. In fact, 5% of the interviewed would never use the system again in case of error. This result contrasts with the answers to the question about which aspect of the system they

considered most important: well functioning, system kindness when it speaks or ease of use. Most said to prefer kindness and ease of use before correct functioning.

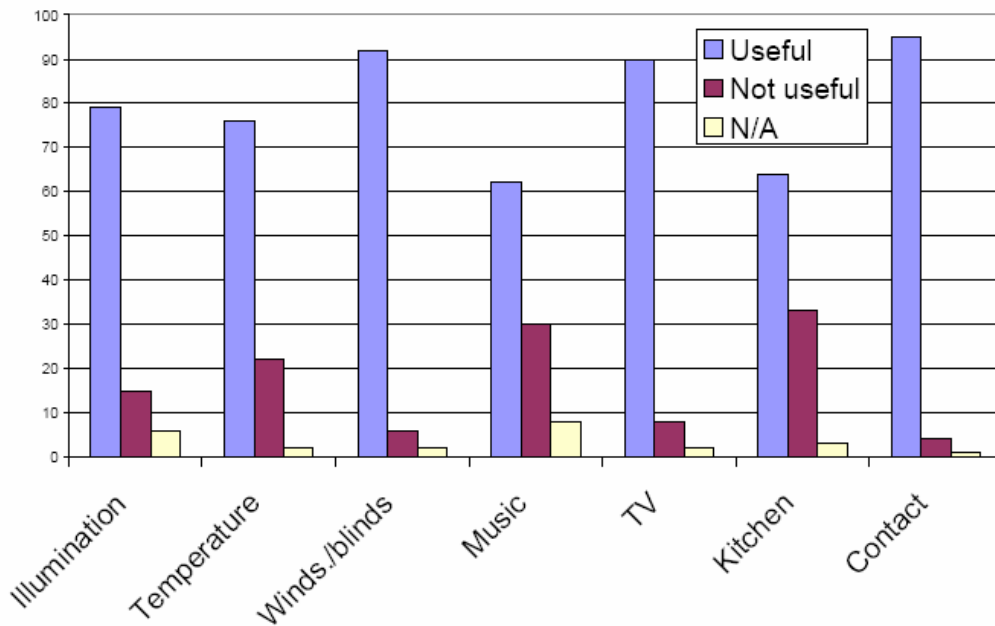


Figure 2. Perceived usefulness of the functionalities.

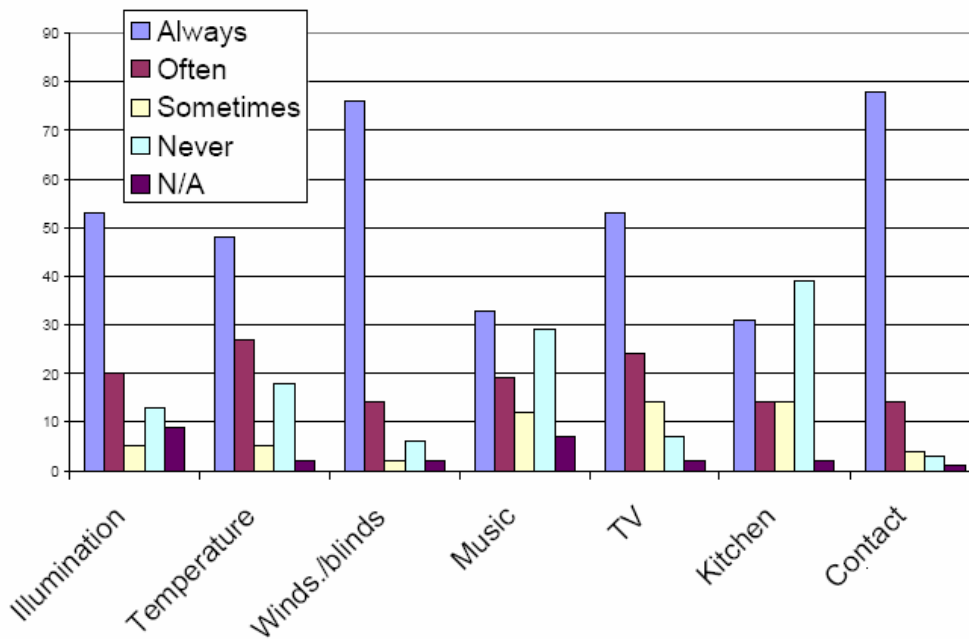


Figure 3. Intended frequency of use of system functionalities.

Finally, when the interviewed were asked to suggest new functionalities, it was surprising to discover their great imagination and how, in despite of their age, they were so open minded to discover totally unexpected pros and cons of the system. For example, related to the illumination control, some suggested that the system could activate an anti-burglaring mechanism when there is nobody home. For example, turning on the lights could make believe that there is someone inside. Others pointed out that the system could evaluate the status of the house and its devices and warn the user if something is not working properly and find automatically somebody to fix the problem (e.g. phone an electrician or plumber). Another feature proposed by some polled is the ability to recognize face and voice not only for detecting the owner but also for recognizing his/her relatives. This way when somebody knocks at the door, the elderly would not have to move if it is not a family member.

When the elderly were interviewed in groups, some of them were able to find solutions to others' fears about this new technology. For example, as said before many of them did not want a human appearance for the system (e.g. an animated agent displayed on the TV screen) because they found it frightening, even when they were told that it could be a cartoon. Someone asked if they would be able to choose the agent face, and when we answered this could be possible, the subject said he would not be afraid if it had the face of his sons or grandsons. The others agreed with him.

Among the disadvantages of the system the youngest indicated that letting it do the work for them could led to loosing their skills sooner due to the lack of use. For example, they said they would loose memory if the system reminds them of all. We think studying this aspect could be a very interesting future work guideline once the system is working.

Conclusions

This paper has presented a new proposal to system design based on three main features: interaction management, user modelling and multimodal interface. This proposal is the result of merging previous strategies towards system specification and design. A very important factor of the new approach is taking the final users' needs and preferences into account right from the initial design states.

As a case of study, the approach has been partially applied to the design of a new multimodal dialogue system to assist elderly people in their daily home activities. The paper has shown how the functionalities and characteristics which a priori seemed to be more useful were not really considered as such by the interviewed potential users (e.g. employing animated agents to enhance the user experience). Moreover, it has shown that the potential users can point out new system features that the designers did not think of, such as automatic recognition of people knocking at the door.

Acknowledgments

This research has been funded by the project HADA TIN2007-64718 of the Spanish Ministry for Education and Science.

References

1. H. Aghajan, J.C. Augusto, R. López-Cózar, *Human-Centric Interfaces for Ambient Intelligence*, Elsevier, 2009.
2. S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J.D. Bolter, M. Gandy, Wizard of Öz Support throughout an Iterative Design Process, *IEEE Pervasive Computing*, November 2005.
3. R. López-Cózar, Z. Callejas, Multimodal Dialogue for Ambient Intelligence and Smart Environments. *Handbook of Ambient Intelligence and Smart Environments*, Springer, 2010.
4. M. Shafto, B. Brown, Human Centered Systems, Proc. AAI 02, Edmonton (Canada), 2002.

About the Authors



Ramon López-Cózar is Professor at the Faculty of Computer Science and Telecommunications of the University of Granada (Spain). His main research interests in the last 15 years include spoken and multimodal dialogue systems as well as Ambient Intelligence. He has coordinated several research projects, has published a number of journal and conference papers, and has been invited speaker at several scientific events addressing these topics. In 2005 he published the book "Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment" (Wiley) and recently has co-edited the book "Human-Centric Interfaces for Ambient Intelligence" (Elsevier Academic Press, 2010). He is a member of ISCA (International Speech Communication Association), FoLLI (Association for Logic, Language and Information), AIPO (Spanish Society on Human-Computer Interaction) and SEPLN (Spanish Society on Natural Language Processing).



Zoraida Callejas is Assistant Professor in the Department of Languages and Computer Systems at the Technical School of Computer Science and Telecommunications of the University of Granada (Spain). She completed a PhD in Computer Science at University of Granada in 2008 and has been a visiting researcher in University of Ulster (Belfast, UK), Technical University of Liberec (Liberec, Czech Republic), University of Trento (Trento, Italy) and Technical University of Berlin (Berlin, Germany). Her research activities have been mostly related to speech technologies and in particular to the investigation of dialogue systems. Her results have been published in several international journals and conferences. She has participated in numerous research projects, and is a member of several research associations focused on speech processing and human-computer interaction.

Development Framework for Pervasive Computing Applications

Václav Slovák, Miroslav Macík, Martin Klíma

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Graphics and Interaction

slovavac@fel.cvut.cz, macikmir@fel.cvut.cz, xklima@fel.cvut.cz

Introduction

Pervasive computing [1] brings the technology closer to the users by enabling the users to use daily-life devices (mobile phones, TVs, touch screen walls, etc.) for controlling their environment and accessing information virtually anywhere. Interacting with such devices does not remind users of classical computers and enables them to more naturally interact with the controlled system, if user interface is designed properly. These devices usually operate in networked environments with every controlling and controlled device connected to a central hub. Bringing easy-to-use applications to such environments faces the challenge of highly heterogeneous, dynamically changing environment and necessity to deploy applications to controlling devices with very different features (display size, input methods, operating systems, etc.).

The *i2home* project is focused on developing smart home applications that would enable various non-technical target groups including elderly people, people with Alzheimer disease, blind people, etc., to take advantage of modern technologies. These technologies would enable the users to stay in touch with their family and be connected to other people while being able to be more independent in daily life. *i2home* tries to achieve this by better integration of home appliances and by providing better easier-to-use controls for the target users.

Recent development in the area of mobile and home entertainment devices led to increased usage of different devices (mobile phones, PDAs, TVs, etc.) for performing various tasks and managing daily life. Such devices may also be used in a smart home for controlling home appliances from any location at home or outdoors. Developing the applications for such environment gets increasingly difficult when different properties of the controlling devices, user preferences/abilities, and different set of controlled appliances are considered. Such constraints force the application developers to either focus on a small subset of available controlling devices and/or to develop multiple interfaces for different platforms that may lead to poor user experience due to inconsistencies across multiple similar products.

Because of the reasons mentioned above, a new development framework was designed for the *i2home* project. This framework is based on the UIProtocol that enables to define interfaces using XML and also defines the standard client-server communication protocol that is very important for pervasive applications in which a server has to handle various client entities that are based on different technologies. In the same time the framework abstracts from details of asynchronous communication between the client and the server. That enables both application logic developers and user interface designers to focus on their primary goals of developing usable and accessible user interfaces and on developing and fine-tuning application logic.

Problem statement

In current environments, the development of the client-side and server-side part of an application is tightly connected with each other and the application logic developer has to communicate often with the user interface designer and vice versa. Often the application logic developers are forced to co-develop user interfaces and user interface designers are forced to implement simple client-side application logic (e.g. in JavaScript). Such development leads to less maintainable code and to a lot of portability issues when changing the client platform. Having server-side application logic simplifies the development and enables to dispatch events, that are invoked by a user's interaction, in a single point on the server side. That enables an easier monitoring processes, debugging the application, and in the future also adding application-wide aspect oriented features [2] that would enable to perform corner-cutting application logic.

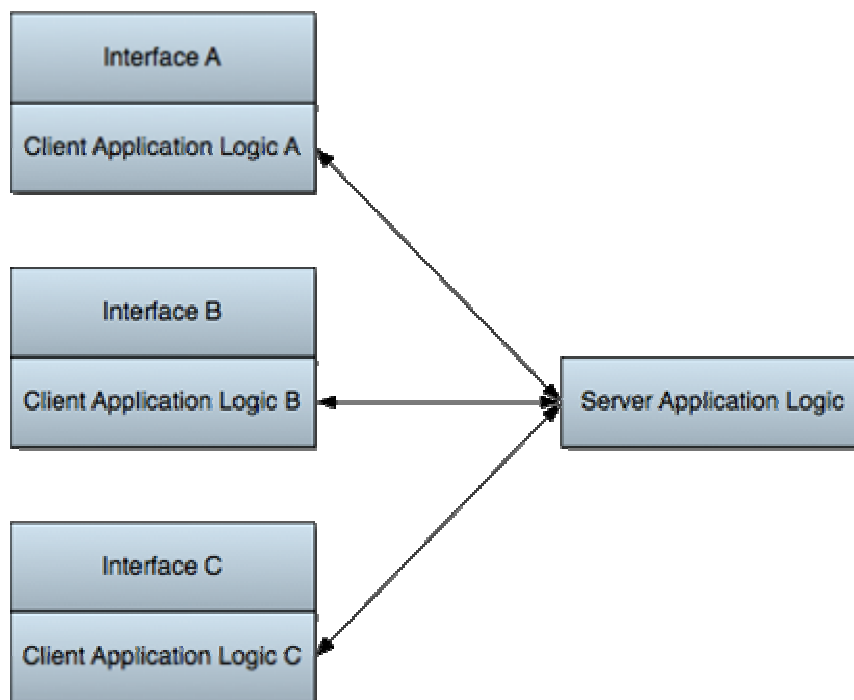


Figure 1: Traditional Client-Server Architecture

The Figure 1 shows how a traditional architecture in pervasive computing environment looks like with each client having different user interface and different client-side application logic. Application logic on client handles communication with the server side and eventually performs some client-side operations as it is closer to user interface. Having a scripting language on the client side is very helpful for creating rich applications with instant feedback to user's actions as there is no latency caused by the network communication. While this is a much-desired feature, it brings a lot of issues related to the efficient development of maintainable applications.

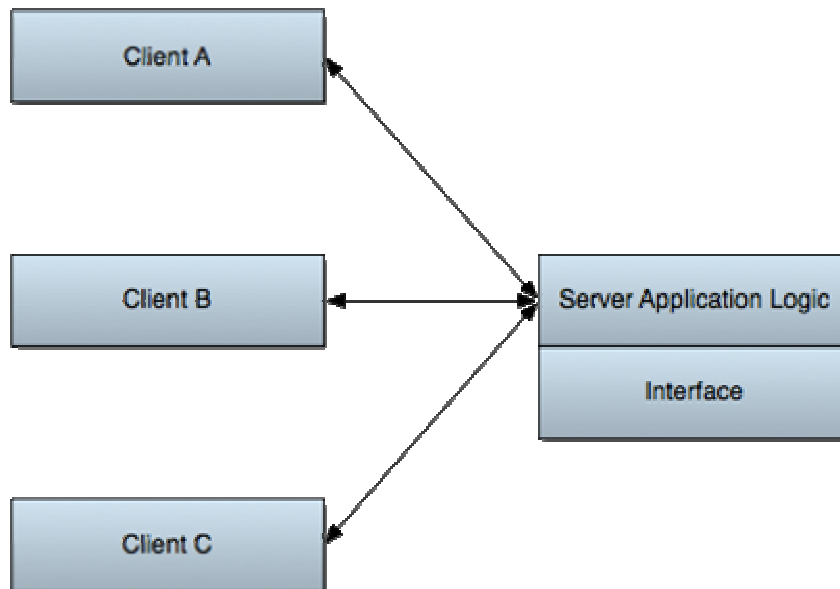


Figure 2: UIProtocol Client-Server Architecture

The Figure 2 shows the way how we intended to minimize the impact of switching platforms on the user interface design and the application logic development: No application logic is implemented on the client side. Even the user interface should be pushed from server in order to simplify deployment similar to most existing web applications. All clients in such framework are thin clients from the developer's point of view. Because this approach requires implementing complex clients (similar to web browsers) it is necessary to ensure that any chosen solution will minimize the complexity of the clients.

In the *i2home* project we followed a User Centered Design. In the iterative design we needed to create a number of prototypes that were immediately verified by usability testing. It is important to rapidly evaluate prototypes on different platforms without a need to rewrite any application logic or a user interface. This enables to evaluate the user interface on multiple devices and choose the best device for users after usability testing is performed. It is also important to be able to create high-fidelity prototypes that may later become fully functional applications without having to re-implement already implemented behavior.

Existing solutions

To date, several various platforms for developing rich internet applications (RIA) are available including but not limited to Adobe Flash, HTML+JavaScript+XML(AJAX), JavaFX and Microsoft Silverlight. The decision to develop UIProtocol framework was based on a study of several existing solutions already available. Adobe Flash was used for developing first prototypes in the *i2home* project. While designing user interface was very fast, creating high fidelity prototypes that would become final applications and would communicate with server side had proven to be difficult with growing complexity of the application.

Apart from the high-profile technologies mentioned above, several other languages has been examined [3, 4] to evaluate whether they are suitable for application development in *i2home* environment.

Adobe Flash

Adobe Flash is the most widely used platform for developing rich internet applications. Adobe Flash is exceptionally powerful tool for creating web vector graphics, animations and delivering media. Adobe Flash provides very good tools for user interface designers, but provides a very poor support for efficient development of the complex application logic. The application logic is often spread thorough the whole user interface and developers have to cooperate very closely in order to complete an application. Debugging the Adobe Flash code is very difficult due to the fact that the application logic is usually spread throughout the debugged application.

Adobe provides Flash development environment and Eclipse-based Flex Builder for developing applications that are more complex. Problematic development of complex applications led several third parties to introduce alternative tools for creating Flash-based applications. These tools however lack some features such as a visual editor.

JavaFX

JavaFX is relatively new solution for rapid prototyping and application development. Stable version of JavaFX 1.2 is available for Mac OS X, Windows and Windows Mobile. Beta version is available for Linux and OpenSolaris. JavaFX enables to built RIA that can utilize very rich Java API. JavaFX behaves similarly to a Java applet or an Adobe Flash movie and may be additionally dragged out of the browser and behave as a standalone application.

JavaFX provides robust support for data binding that proves very useful when dealing with the user interfaces. The major disadvantage is the fact that the application logic is still on the client side [5] and implementation of JavaFX client is relatively complex as whole Java Virtual Machine must be available and support for non-XML JavaFX syntax has to be implemented.

Several development tools are available for JavaFX enabling to import Adobe Photoshop and Adobe Illustrator files and add behavior to the static content designed in these applications.

HTML and JavaScript

HTML is an open standard that can be rendered on almost any platform including mobile devices. The look and feel of any HTML content may be customized by CSS, while dynamic behavior may be added by JavaScript. Communication with server is done by AJAX [6] that is used to exchange the XML (or other files) in the background dynamically updating only portions of a website bringing HTML close to other RIA technologies. For the user interface updates to appear instant, a JavaScript must be invoked in a defined intervals to ask the server for updates.

While the HTML with CSS and JavaScript in theory offer features that match other RIA technologies, the applications get much more complex and less maintainable, due to a lot of application logic written in a scripting language on the client side. For developing RIA applications, the HTML is often mixed with Adobe Flash because the development in pure HTML gets more difficult for highly interactive applications.

There are many development tools for HTML and related technologies that enable both visual editing and direct code editing. The key problem remains debugging.

Google Web Toolkit

Google Web Toolkit (GWT) is based on an idea similar to UIProtocol in terms of moving all application logic to server side. It uses Java to JavaScript translator that enables to write application logic in Java programming language having all the advantages of statically typed language while developing client-side application logic. For the client side logic there is only a subset of Java API that can be translated to JavaScript available. For the server side application logic it is possible to use full set of Java API. User interfaces in GWT are created programmatically using Java language. Parts of user interfaces generated by GWT may be embedded into static HTML websites and may also use CSS styles for modifying look and feel of the user interface. All user interface components in GWT provide methods that enable directly setting HTML-related properties such as CSS style.

The main disadvantages are that the user interface is created programmatically. While creating user interfaces programmatically is very flexible it is not natural for user interface designers. Application logic running on client and server may also be mixed with each other and also with creating a user interface.

During development GWT applications are run in a special environment (hosted mode) that enable to debug the application directly in a Java IDE overcoming some shortcomings of JavaScript debugging and accelerate development. After debugging in the hosted mode the application is evaluated directly in web browsers to solve browser-specific problems.

GWT is supported in all the major Java IDEs including Eclipse, IntelliJ IDEA and Netbeans.

Microsoft Silverlight

Microsoft Silverlight [7, 8] technology is Microsoft's competitor to currently dominating Adobe Flash. It is based on Microsoft .NET Compact Framework and aims to provide similar features to Adobe Flash while making the platform more developer friendly. Microsoft Silverlight enables to write application code in any .NET compatible language, but API is limited to .NET compact framework due to security concerns and dependency of .NET framework on native Microsoft Windows-only libraries.

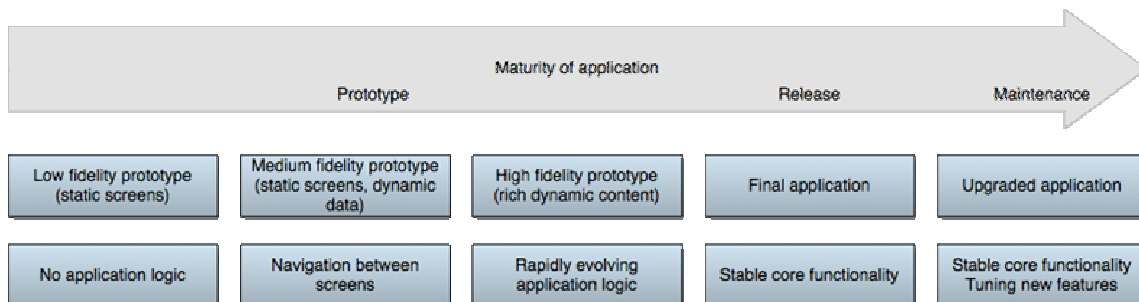


Figure 3: Application Development

Development Process

The major motivation behind UIProtocol was to create a solid foundation for rapid prototyping of user interfaces and in the same time enabling the prototypes to be converted to fully functional applications without additional development. In the Figure 3 is depicted the typical development process of an application. First a set of prototypes with incrementally increased degree of fidelity is created. A test results from usability testing of a prototypes are employed in a next prototype. Increasing the fidelity of the prototypes leads to implementing a lot of application logic that gets debugged in the process. Not using the already implemented application logic leads to wasting a lot of resources and opens a possibility to reintroduce already solved problems.

UIProtocol enables to simply cover all phases of the development and maximize usage of already developed solutions. Ability to develop application logic in both scripting (e.g. ECMAScript) and static languages (e.g. Java) enable to rapidly evaluate and alter application logic written in scripting language and then eventually implementing it in a static language as it becomes stable.

Another focus is to enable simple adding of the dynamic behavior to a static content of a low fidelity prototype. This is achieved by binding support that enables the user interface designer to simply identify dynamic content in a prototype and also enables application logic developer to update the dynamic data without knowing how they are exactly represented in the user interface. The binding simply connects the dynamic application data with their representation in a user interface. Whenever the data are updated, all user interface components that are influenced by this change are updated automatically.

Architecture

UIProtocol was designed for the client-server architecture that is typical for pervasive applications by clearly separating the client and the server part. Still, both client and server may run on a single machine, behaving like a standalone application. This enables application developers to choose how the application will be deployed very late in the application development lifecycle.

The client in the UIProtocol architecture is a terminal device that is used by the user to interact with the system and is defined by the following properties:

1. Insecure environment (a device running client may be stolen or abused by user to threaten server or be used to send malformed data to server or the communication link between client and server may be attacked)
2. Directly receives a user's input
3. Renders interface for a user
4. Not allowed to perform any critical application logic

The server in UIProtocol architecture is defined as the following:

1. Running on a trusted machine (thus the application can directly access data, check user input, etc.)
2. Performing critical application logic and manipulating persistent data
3. Allowing multiple clients to be connected in the same time and ensuring proper information exchange between clients (synchronization and transactions)

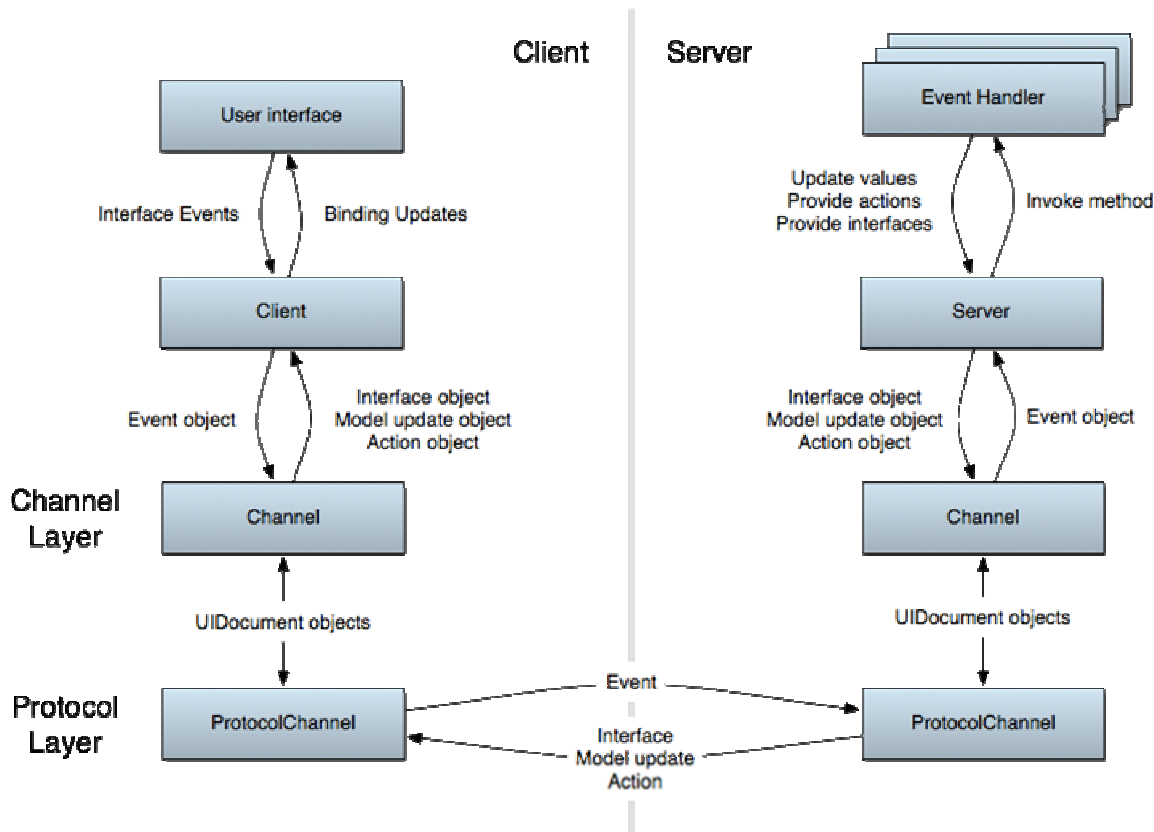


Figure 4: UIProtocol Client-Server Architecture

A UIProtocol server provides the client the interfaces that should be rendered and dynamic data (application-wide models) that should be displayed in the user interface. The client is allowed to send only events to server without actually knowing what the events will cause.

The communication and type of information that are exchanged between client and server is shown in the Figure 4 that describes current client and server implementation in *i2home* project.

Separation of UI and Application Logic

One of the most important features of the UIProtocol is that it does not enable any application logic to be used on the client side. Although an implementation of a server may offload some application logic to the client, this process should be totally transparent from the developer's point of view. No imperative (scripting-like) language features are enabled by design in the UIProtocol.

The main purpose of separation of the application logic and the user interface was to minimize the amount of information exchanged between the user interface designer and the application logic developer.

The information that is exchanged between the user interface designers and the application logic developers is limited to identifiers of events that are send from the client (and their properties) and to meaning of values in application models containing the

dynamic data. This information is usually provided by the user interface developer, as he/she usually identifies what data in a user interface should be dynamic and what application logic should be executed when the user interacts with the user interface.

Binding

Binding is essential for separating data from the user interface in the UIProtocol client. Thanks to the data binding, the server-side application logic does not have to directly access the user interface structure to modify the displayed content (similarly to DOM [9] in HTML+JavaScript). All user interface components displaying the dynamic data are automatically updated when the data are changed. Additionally all dynamic data can be interpolated to a newly assigned value. This enables to use the animations very easily virtually anywhere in the user interface, thus improving user experience.

Pushed Updates

The UIProtocol clients also provide support for pushed updates as there is permanently opened socket between client and server. This enables an immediate reaction of the user interface to any activity on the server side. While keeping socket open requires some resources to remain allocated, the communication is then faster as it is not necessary to re-establish connection for every request. Furthermore, the server does not have to maintain the information it receives from an external source until all clients ask for updates, but may automatically push that information to all affected clients.

Protocol Abstraction Layer

Although currently UIProtocol is used for exchanging user interface-related information the architecture itself is not strictly bound to using UIProtocol language. Any language that has the same expressing power may be used. Each UIProtocol document is a human readable XML file that makes it easy to edit by any XML editor. However that makes UIProtocol very memory inefficient. In current environment we also use UIProtocol-B which is a binary representation of UIProtocol that is faster to process and requires less bandwidth. These features may be especially important in pervasive computing environments as mobile devices have less processing power and data transfers over wireless are usually much slower than wired connections.

As a visual editor for UIProtocol is under development we expect that the human-readable UIProtocol won't be necessary in the future and more efficient language may be used instead.

Simple Client Structure

UIProtocol was designed to be easy to process and easy to implement. No necessity to implement a scripting engine, and concept of binding used thorough the client enables simply wrap available native user interface components. There is also no necessity to implement all features of UIProtocol as some of the features such as animations and complex components may not be supported by client in order to keep user interface functional.

Language Agnostic

UIProtocol was developed to enable all application logic to be written in statically typed languages, because they enable to write more error-free code due to advanced tools available in current IDEs such as static code analysis. By design UIProtocol does not require

the application logic to be written in any particular language and the event handlers in a single application may even be written in different languages. Current Java server implementation enables event handlers to be written in Java and any scripting language supported by Java Scripting API including ECMAScript, Groovy, Python, Ruby, etc. Support for additional format of event handlers may be added in the future without having any impact on server architecture.

Using several different languages for handling events may seem to defeat the purpose of simplifying the development and make it more error-free, especially when using dynamic scripting languages for handling events. Using the scripting languages enables to alter the behavior of the server-side quickly during debugging and testing without having to recompile and redeploy the application. Additionally, using languages such as ECMAScript is simpler for less experienced developers. Writing portions of rapidly evolving application logic in a scripting language and migrate to native handlers after e.g. usability testing may significantly accelerate development as these logic may have to be modified several times before reaching mature status.

Implementation

UIProtocol Client

Currently, several implementations of UIProtocol are available with different level of compatibility with the current specification, due to the fact that UIprotocol is still under development. Currently two most advanced implementations of clients are available for Java and Microsoft .NET platforms. There are two additional client implementations for Adobe Flash and Microsoft Silverlight.

Figure 5 shows example user interfaces designed for a living room TV using .NET UIProtocol client. Figure 6 shows user interfaces designed for touch screen device that is located near doors and enables user to check status of household and control a home security system.

UIProtocol Proxy

UIProtocol proxies enable to connect clients that do not support UIProtocol to UIProtocol server. They provide abstraction level for UIProtocol server because they behave as an ordinary client while translating all events, interfaces, model updates and actions to a format that is understandable by a proxied client. Figure 7 shows a proxy that translates UIProtocol to HTTP so a web browser may be used to connect to a UIProtocol server. An implementation of such proxy in ASP.NET is available providing support for basic UIProtocol features. In future versions of servers this functionality may be directly integrated.

UIProtocol Server

We have developed 2 UIProtocol servers in *i2home* project. The server currently used in *i2home* environment is based on Microsoft .NET platform and fully supports UIProtocol except streaming, that might be used in the future for transferring VOIP data, upload files, etc.

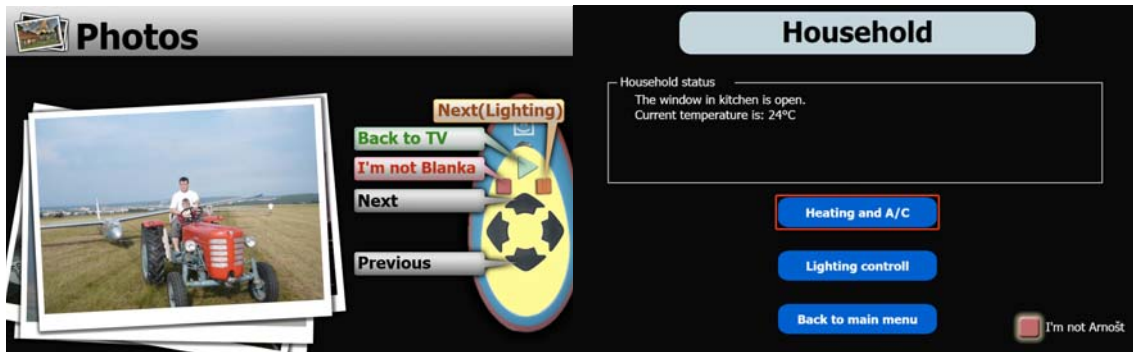


Figure 5: Interfaces rendered by .NET client

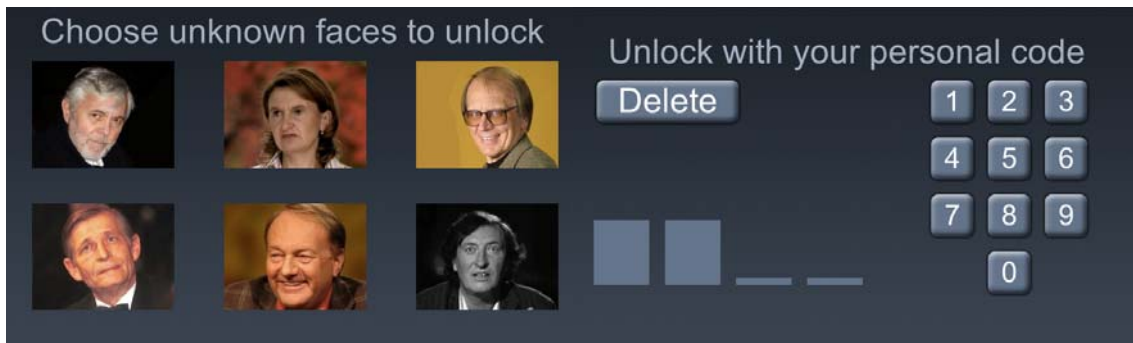


Figure 6: Interfaces rendered by Java client

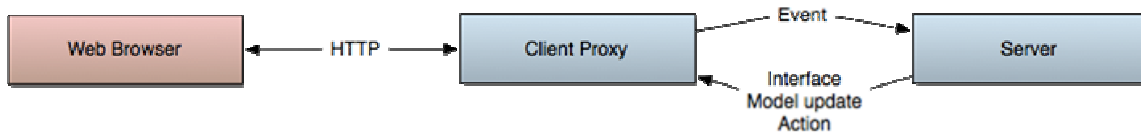


Figure 7: HTTP-UIProtocol proxy

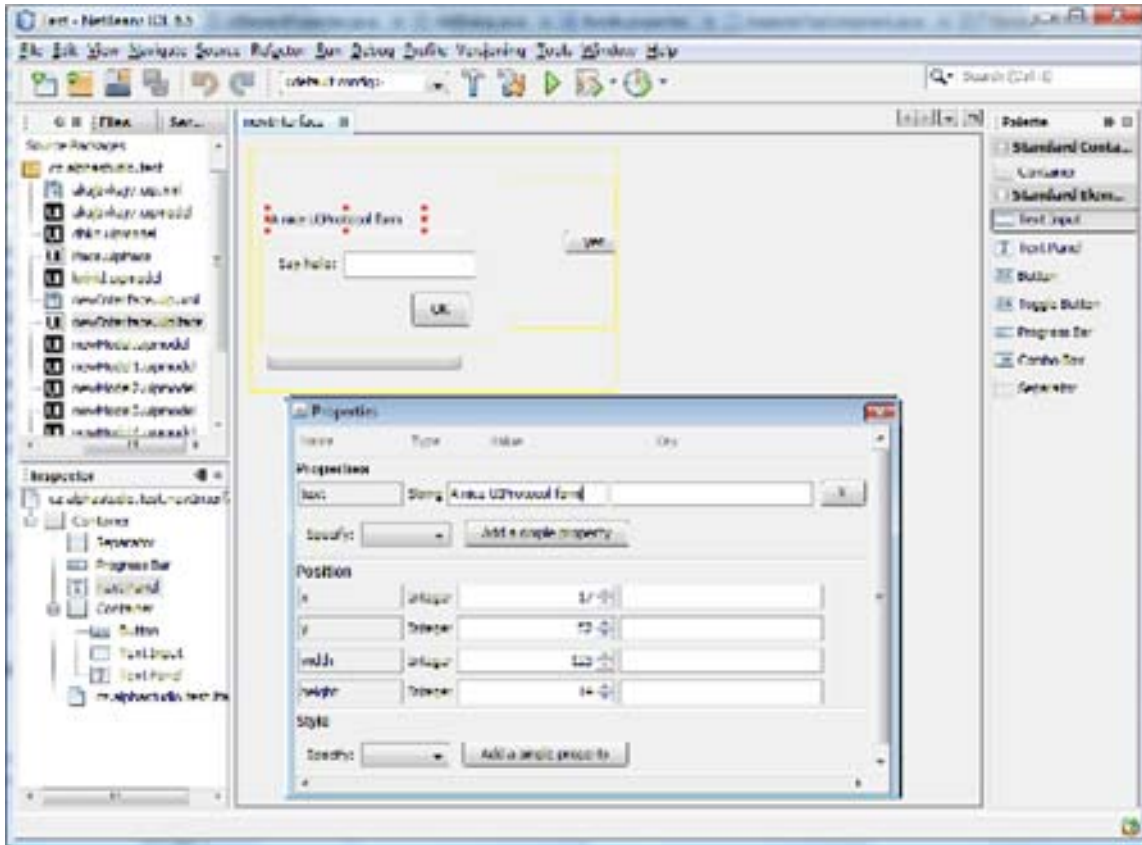


Figure 8: UIProtocol Editor

UIProtocol Editor

A visual editor (figure 8) for UIProtocol is currently under development. The editor is based on the open source Netbeans platform and will be available as a plugin for the Netbeans IDE. Currently, the editor supports drag and drop of user interface components, modifying properties of the components, positioning them, simply binding user interface components to dynamic data and attaching behavior to the components.

Conclusion

The UIProtocol draft specification is available and defines concepts and architecture of the development framework discussed in this paper. The current specification also includes support for basic user interface components, their properties, styling, positioning and layout. The final list of supported elements is not finalized, but serves well for creating user interfaces in *i2home* project. It is necessary that real-world scenarios are evaluated in order to finalize list of components and their properties.

Future Research

Future research will be focused on finalizing UIProtocol specification by strictly defining behavior for each user interface component.

As UIProtocol architecture enables implementation of application logic in any language and even use any combination of languages in a single application. Using workflows and task models may be considered for modeling high-level application logic and complex processes inside an application.

Acknowledgments

This research has been done within project *i2home* funded by the sixth Framework Program of European Union under grant FP6-033502 (*i2home*). This research has been partially supported by the Ministry of Education, Youth, and Sports of the Czech Republic under the research program MSM 6840770014. This research has been also partially supported by the Ministry of Education, Youth, and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics).

Bibliography

1. Weiser, M., *Ubiquitous computing*. IEEE Computer, 1993. **26**(10): p. 71-72.
2. Kiczales, G., et al., *Aspect-oriented programming*. 2002, Google Patents.
3. Souchon, N. and J. Vanderdonckt, *A review of xml-compliant user interface description languages*. Lecture notes in computer science, 2003: p. 377-391.
4. Theses, M. and G. Abroad, *A Platform-Independent User Interface Description Language*.
5. Weaver, J., *JavaFX Script: Dynamic Java Scripting for Rich Internet/Client-Side Applications*. 2007: Apress.
6. Eichorn, J., *Understanding AJAX: Using JavaScript to Create Rich Internet Applications*. 2006: Prentice Hall PTR.
7. MacVittie, L.A., *XAML in a Nutshell*. 2006: O'Reilly Media, Inc.
8. Swift, J., et al., *Professional Silverlight 2 for ASP.NET Developers*. 2009: Wrox Press Ltd.
9. Wood, L., et al., *Document object model (dom) level 1 specification*. W3C recommendation, REC-DOM-Level-1-19981001.

About the Authors



Václav Slováček is a PhD student at Czech Technical University (CTU) in Prague. He received his master degree in software engineering at CTU in 2009. He is currently involved in *i2home* and VITAL projects that are funded by 6th Framework Program of European Union. His research is focused on development of accessible user interfaces by using task models early in application development lifecycle to optimize workflows in application for different user groups.



Miroslav Macík is a PhD student and junior researcher at the Czech Technical University in Prague, Department of Computer Graphics and Interaction. He graduated the same university in 2009 receiving his master's degree. His Master's thesis "User Interface Generator" is followed by his research where he focuses on automatic user interface generation. He is further interested in usability engineering, accessibility and web technologies. Currently he is involved in two EU founded projects *i2home* and VITAL (6th Framework program).



Martin Klíma is a researcher and a lecturer at the CTU in Prague, Department of Computer Graphics and Interaction. He received his PhD in 2009 at the same university. His research interests include mobile user interaction, data adaptation for mobile environment, computer supported collaborative work in mobile environment, and user interfaces for users with special needs like elderly or handicapped. Between 2001 and 2002 he was working on adaptation of multimedia documents on PDA at ZGDV Darmstadt, Germany (analysis, design, prototype implementation). After his return to Prague he became involved in number of EU funded projects including Mummy, ELU, *i2home*, Vital, Vital Mind, Aegis and others. He is an author or a co-author of more than twenty publications on various international events in HCI.