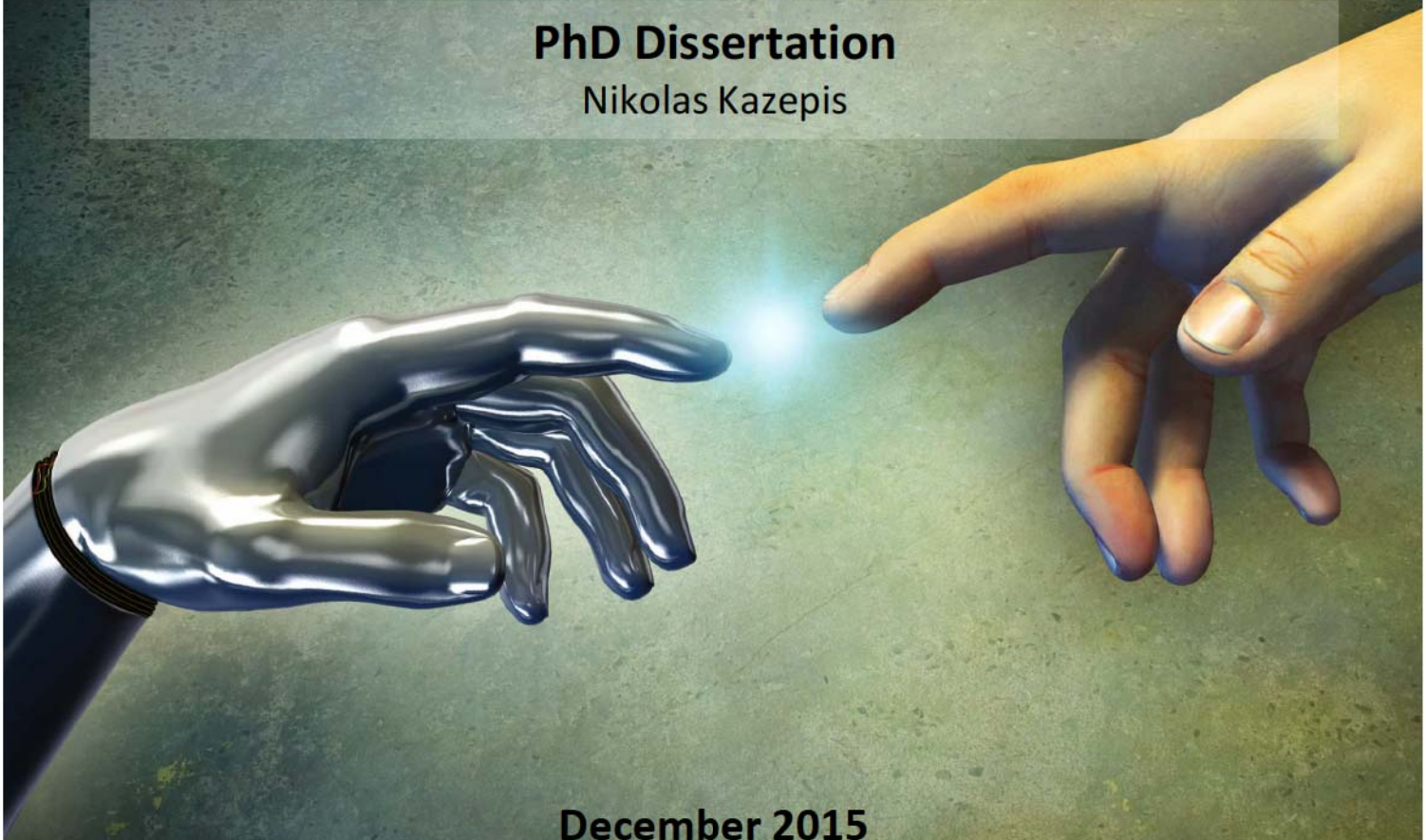
FIRMA: A Development Framework for Elderly-Friendly Interactive Multimodal Applications for Assistive Robots

PhD Dissertation

Nikolas Kazepis



December 2015

FIRMA: A Development Framework for Elderly-Friendly Interactive Multimodal Applications for Assistive Robots

Ph.D. Dissertation

Nikolas Kazepis

HERAKLION, DECEMBER 2015

University of Crete
Department of Computer Science

FIRMA: A Development Framework for Elderly-Friendly Interactive Multimodal Applications for Assistive Robots

Dissertation submitted by Nikolaos Kazepis in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science.

Doctoral Thesis Committee:	Constantine Stephanidis, Professor, (Advisor)		University of Crete
	Apostolos Traganitis, Emmeritus Professor,		University of Crete
	Antonis A. Argyros, Professor,		University of Crete
	Xenophon Zabulis,	Principal Researcher	ICS FORTH
	Dimitris Grammenos,	Principal Researcher,	ICS FORTH
	Margherita Antona,	Principal Researcher,	ICS FORTH
	Dimitrios Tzovaras,	Principal Researcher,	ITI CERTH

The work reported in this thesis has been conducted at the Human Computer Interaction (HCI) laboratory of the Institute of Computer Science (ICS) of the Foundation for Research and Technology – Hellas (FORTH), and has been financially supported by a FORTH-ICS scholarship. Additionally, this work has been supported by the FORTH-ICS internal RTD program "Ambient Intelligence Environments". Finally, part of this work has been conducted in the context of the project RAMCIP -Robotic Assistant for MCI Patients at home- which is an EU Horizon 2020 funded project, under the grant agreement no: 643433. The project started on the 1st of January 2015 and has a duration of 36 months.

University of Crete
Department of Computer Science

FIRMA: A Development Framework for Elderly-Friendly Interactive Multimodal Applications for Assistive Robots

Dissertation submitted by Nikolaos Kazepis in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science.

Author: Nikolaos Kazepis

Examination Committee: Constantine Stephanidis, Professor, University of Crete

Apostolos Traganitis, Emeritus Professor, University of Crete

Antonis A. Argyros, Professor, University of Crete

Xenophon Zabulis, Principal Researcher, ICS FORTH

Dimitris Grammenos, Principal Researcher, ICS FORTH

Margherita Antona, Principal Researcher, ICS FORTH

Dimitrios Tzovaras, Senior Researcher (Researcher A'), Director of the Informatics and Telematics Institute (ITI) at the Centre for Research and Technology Hellas (CERTH).

Department Approval: Panagiotis Tsakalides, Professor, Head of the Department

Heraklion, October 2015

I dedicate this dissertation to my loving father Ioannis and my late loving mother Despoina for making me be who I am, for tolerating me and supporting me all the way and for inspiring me, guiding me and helping me through thick and thin in my endeavors.

Sparkle and shine in the face of darkness, you two light up the shadows

Declaration of Originality

I declare that this thesis is my own work. Information derived from published or unpublished work of others has been formally acknowledged.

Acknowledgements

I would like to thank my thesis supervisor Professor Constantine Stephanidis and Dr. Margherita Antona for their strong support, encouragement and assistance through my Ph.D. studies. They helped me to improve my research skills with their close attention to my work and critical thinking while their ideas, guidance and support were invaluable in order to shape this work to its current state.

I am also grateful to Professor Antonis Argyros, Dr. Dimitris grammenos and Emmeritus Professor Apostolis Traganitis for their invaluable insights, suggestions and comments, which also provided clear guidance towards exploring interesting future directions of my work.

A very special acknowledgement goes to my beloved girlfriend Mandy Milonaki for being my source of inspiration. Her contribution in terms of tolerating me and providing me with all the psychological support and the courage required to make this work happen was critical. I would like to thank her for all her love and support especially during the final, critical months of my dissertation and for making me feel like anything was possible.

Moreover I would like to thank in particular my friends and colleagues Nikolaos Partarakis and Manos Zidianakis for their valuable help, support, contribution and encouragement throughout this work.

I would like to say a big thank you to my friend and colleague Antonis Katzourakis for his superhuman effort in providing me with high quality designs and graphics for various aspects of the UIs under extremely strict deadlines.

I would also like to thank my colleague Danae Ioannidi for her help, support and scientific insights.

In addition, I would also like to thank the Department of Computer Science of the University of Crete and the Institute of Computer Science (Human Computer Interaction Laboratory) of the Foundation for Research and Technology – Hellas (FORTH) for providing me with the means for setting and achieving my research goals.

A big thank you to all my friends there, who supported me and my work and helped in making the office environment stimulating, productive and pleasant. In particular, I would like to thank my friends and colleagues Kostas Pappoutsakis and Ammar Qammaz at the CVRL lab of ICS-FORTH for their help and support regarding the computer vision parts of this work.

Finally, I need to thank my parents Ioannis and Despoina, my foster mother Eleni, my sister Frantzeska (with an 'n') and my friends for supporting and tolerating me all these years.

To all of the above, a huge THANK YOU. I wouldn't have succeeded without you.

Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών
Σχολή Θετικών και Τεχνολογικών Επιστημών

Ένα Προγραμματιστικό Πλαίσιο Ανάπτυξης για Φιλικές ως προς τους Ηλικιωμένους, Πολυτροπικές, Προσαρμοζόμενες, Διαδραστικές Εφαρμογές για Οικιακές Υποστηρικτικές Ρομποτικές Πλατφόρμες

Διδακτορική διατριβή που υποβλήθηκε από το Νικόλαο Καζέπη

Περίληψη

Η προοδευτική αύξηση του προσδόκιμου όρου ζωής και η επακόλουθη αύξηση του πληθυσμού της τρίτης ηλικίας στην Ευρώπη και σε παγκόσμιο επίπεδο επιφέρουν την ανάγκη για νέες τεχνολογικές λύσεις στον τομέα της βελτίωσης της υγείας, της ανεξάρτητης διαβίωσης, της ποιότητας ζωής και της ενεργού γήρανσης των πολιτών της Κοινωνίας της Πληροφορίας. Η ποιότητα ζωής και η ενεργής γήρανση των ηλικιωμένων έχει γίνει ένας σημαντικός στόχος των ευρωπαϊκών κοινωνιών σήμερα. Στο πλαίσιο αυτό, ο όρος «Υποβοηθούμενη Αυτόνομη Διαβίωση» (Ambient Assisted Living - AAL) αναφέρεται στην ουσιαστική χρήση των τεχνολογιών, με τέτοιο τρόπο ώστε να βελτιώνουν την ποιότητα ζωής των ηλικιωμένων, προσφέροντάς τους την ασφάλεια και την άνεση που χρειάζονται για να ζήσουν ανεξάρτητοι και για περισσότερο χρόνο στο σπίτι τους, σε αντίθεση με τον εγκλεισμό τους σε κάποιο γηροκομείο ή ίδρυμα.

Πολλές μελέτες έχουν διεξαχθεί στον τομέα της υποστήριξης των ηλικιωμένων σε περιβάλλοντα υποβοηθούμενης διαβίωσης, όσον αφορά τη διευκόλυνσή τους στις διάφορες πτυχές της καθημερινής τους ζωής, στην αντιμετώπιση ιατρικών περιστατικών και στη λήψη κατάλληλων μέτρων σε περιπτώσεις έκτακτης ανάγκης.

Ωστόσο, εξίσου σημαντική πτυχή στη ζωή των ηλικιωμένων είναι η ενεργός συμμετοχή τους στο περιβάλλον διαβίωσής τους και αυτό συνεπάγεται την αλληλεπίδρασή τους με διάφορους και διαφορετικούς τύπους τεχνολογιών. Ένας τέτοιος τύπος τεχνολογίας είναι οι οικιακές υποβοηθητικές ρομποτικές πλατφόρμες.

Οι υποβοηθητικές ρομποτικές πλατφόρμες βρίσκονται στο προσκήνιο εδώ και αρκετό καιρό, στο πλαίσιο της προσπάθειας των ερευνητών να ξεπεράσουν ουσιαστικά προβλήματα που σχετίζονται με τη ρομποτική τους φύση και τη χρήση τους σε οικιακά περιβάλλοντα. Η επιτυχής ένταξη υποβοηθητικών ρομπότ σε οικιακά περιβάλλοντα υπήρξε το αποτέλεσμα διεπιστημονικών προσπαθειών από διάφορους επιστημονικούς τομείς που κυμαίνονται από τον τομέα της ρομποτικής και υπολογιστικής όρασης μέχρι τον τομέα της μηχανικής μάθησης και των πληροφοριακών συστημάτων. Οι προκλήσεις που έχουν κληθεί να αντιμετωπίσουν αυτές οι προσεγγίσεις είναι τόσο μεγάλες ώστε περισσότερη έμφαση έχει δοθεί στην επίτευξη της ασφαλούς συνύπαρξης των ρομπότ σε οικιακά περιβάλλοντα παρά στο θέμα της αλληλεπίδραση μεταξύ της ρομποτικής πλατφόρμας και των χρηστών της. Ωστόσο, δεδομένου ότι το πεδίο της ρομποτικής έχει ωριμάσει αρκετά τα τελευταία χρόνια, η εστίαση στην αλληλεπίδραση μεταξύ των ανθρώπων και των ρομπότ σε οικιακά περιβάλλοντα υποβοηθούμενης αυτόνομης διαβίωσης γίνεται όλο και πιο αναγκαία.

Η εργασία αυτή εστιάζει στην αλληλεπίδραση και τη συνεργατική συνύπαρξη των ηλικιωμένων και των ρομπότ. Ο κύριος στόχος είναι η επίτευξη μιας γλώσσας αλληλεπίδρασης η οποία θα είναι από κοινού κατανοητή. Σε αυτή την νέα μορφή επικοινωνίας, η αλληλεπίδραση θα πρέπει να είναι προσαρμοσμένη στους τελικούς χρήστες, λαμβάνοντας υπόψη τις ειδικές απαιτήσεις του κάθε ατόμου, την κατάσταση του περιβάλλοντος, αλλά και τις δυνατότητες που παρέχονται από τις ρομποτικές πλατφόρμες. Προς αυτόν το σκοπό, η παρούσα διατριβή επικεντρώνεται στη δημιουργία μιας καθολικής λύσης, η οποία θα μπορέσει να χρησιμοποιηθεί ως ακρογωνιαίος λίθος για την ανάπτυξη πολυτροπικών, φιλικών ως προς του ηλικιωμένους, διαδραστικών εφαρμογών που προορίζονται για ρομπότ οικιακής χρήσης. Επιπλέον, η ερευνητική αυτή εργασία παρέχει στους προγραμματιστές τις απαραίτητες τεχνολογίες, τα εργαλεία και τα δομικά στοιχεία που χρειάζονται για τη

δημιουργία εύχρηστων, φιλικών ως προς του ηλικιωμένους, πολυμεσικών εφαρμογών σε AAL περιβάλλοντα, με ιδιαίτερη έμφαση στις ρομποτικές πλατφόρμες, αυξάνοντας έτσι τις δυνατότητες προσαρμογής τους στις ανάγκες των χρηστών και κατά συνέπεια την αποδοχή των τεχνολογιών αυτών από τους τελικούς χρήστες. Οι εφαρμογές που αναπτύσσονται με τη χρήση του προτεινόμενου πλαισίου είναι εγγενώς φιλικές ως προς τους ηλικιωμένους, ενώ παράλληλα μπορούν και προσαρμόζονται στις ανάγκες τους, στο περιβάλλον και στο πλαίσιο χρήσης. Με αυτόν τον τρόπο παρέχεται μια ομαλή καμπύλη μάθησης και ταυτόχρονα αυξημένα επίπεδα ικανοποίησης στους τελικούς χρήστες. Επιπλέον, η χρήση του προτεινόμενου πλαισίου εισάγει νέους τρόπους αλληλεπίδρασης, όπως η αλληλεπίδραση με χρήση φωνής και χειρονομιών, εμπλουτίζοντας έτσι την εμπειρία των τελικών χρηστών και συνάμα απλοποιώντας την αλληλεπίδραση και το χειρισμό των εφαρμογών. Τέλος, το πλαίσιο διευκολύνει την αποτελεσματική και αποδοτική ανάπτυξη διεπαφών χρήσης, απλοποιώντας έτσι σε μεγάλο βαθμό το έργο του προγραμματιστή.

Abstract

The continuous growth of the older population in Europe and the progressive ageing of society worldwide bring about the need for new technological solutions for improving the health, independent living, quality of life and active ageing of older citizens in the Information Society. Quality of life and active aging of elderly people is becoming an essential objective for today's European societies. In this scope, the term "Ambient Assisted Living" (AAL) is used to refer to the meaningful usage of information and communication technologies, in a way that they can improve the quality of life of older people by offering them safety and comfort to live longer at home and prologue their independence. Many studies have been carried out about empowering the elderly in AAL environments, concerning the aspects of facilitating everyday living, addressing medical conditions and taking appropriate actions in emergency situations. However, an equally important aspect of the elderly's lives is their active engagement in their surrounding environment and this implies interacting with different types of technology.

Robotic platforms have been around for quite some time since researchers have been trying to overcome essential problems that are related to the nature of robotics and their usage in domestic environments. Successful integration of robotic platforms in a domestic environment has been the result of multidisciplinary efforts from various scientific fields ranging from computer vision to machine learning and information systems. The challenges of these approaches have been so overwhelming that more focus was given to achieving a safe robot existence in a domestic environment than the actual human-robot interaction between the platform and its users. However, since the field of robotics has matured over the last years, a focus shift from the hardware itself to the HRI part of the robot's existence in such domestic environments is becoming more and more necessary.

Furthermore research efforts have focused on incorporating household robotic platforms in such environments under the role of domestic care givers or social companion. The possibilities that are offered by these domestic platforms towards the assistance in everyday life and the wellbeing of the senior citizens are being

extensively researched in recent efforts as this scientific field has been drawing considerable attention in recent years. To this end, domestic robotic platforms have been proved to be a valuable piece of technology since they can actively and proactively assist senior citizens in their everyday lives.

This work provides a focus shift towards the interaction part of the collaborative co-existence of the elderly and the robot. The main goal is to achieve the development of a commonly understood interaction language. In this novel form of communication, interaction should be tailored to the end users taking into account the specific requirements of each individual, the environmental state but also the capacity of the input/output channels provided by the robotic platform. To this end, the current thesis focuses on creating a universal solution to be used as the corner stone for building multimodal, elderly friendly, interactive applications that target household robots for elderly users. This research work provides developers with the necessary technologies, tools and building blocks for creating easy to use elderly-friendly multimodal applications in AAL environments, with particular focus on robotic platforms, thus increasing their level of adaptation to users' needs, and, as a consequence, users' acceptance of these technologies. The applications developed using the proposed framework are inherently friendly to the elder users, while adapting to their needs, the surrounding environment and the context of use. This results in a smooth learning curve, providing increased levels of user satisfaction. Moreover, the use of the proposed framework introduces new interaction modalities such as voice and gestures, enriching and simplifying the interaction experience of the applications. Finally, the framework facilitates effective and efficient development of the supported user interfaces, thus simplifying to a great extent the developer's work.

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

HRI	**H**uman **R**obot **I**nteraction
UI	**U**ser **I**nterface
GUI	**G**raphical **U**ser **I**nterface
AmI	**Am**bient **I**ntelligence
ROS	**R**obot **O**perating **S**ystem
U2I	**U**nified **U**ser **I**nterfaces
XAML	e**X**tensible **A**pplication **M**arkup **L**anguage
XML	e**X**tensible **M**arkup **L**anguage
ACTA	A general purpose finite state machine (FSM) description language for **ACT**ivity **A**nalysis
FSM	**F**inite **S**tate **M**achine
RAMCIP	**R**obotic **A**ssistant for **M**ild **C**ognitive **I**mpairment **P**atients
AAMI	Age Associated Memory Impairments
GSR	**G**alvanic **S**kin **R**esponse
IVR	**I**nteractive **V**oice **R**esponse
DTMF	**D**ual-**T**one **M**ulti-**F**requency signaling
SRGS	**S**peech **R**ecognition **G**rammar **S**pecification
SAPI	**S**peech **A**pplication **P**rogramming **I**nterface
VUM	**V**irtual **U**ser **M**odel
WinFX	**Win**dows **F**ramework e**X**tension
SSML	**S**peech **S**ynthesis **M**arkup **L**anguage
ARMA	**A**CTA **R**untime for **M**ultimodal **A**pplications
MCI	**M**ild **C**ognitive **I**mpairment

1 Introduction

1.1 Defining 'Old Age'

Old age can be defined as ages nearing or surpassing the average life span of human beings, and thus the end of the human life cycle. Euphemisms and alternative terms for old people include seniors (American usage), senior citizens (British and American usage) and the elderly.

The boundary between middle age and old age cannot be defined exactly, because it does not have the same meaning in all societies. People can be considered old because of certain changes in their activities or social roles (i.e., when they become grandparents, or when they retire from work).

Defining the "old age" is not quite as obvious as it might seem. Much of the recent interest in designing for elders has been motivated by the idea that the world's population is 'ageing'. Implicit in the notion of the ageing society is the idea that "old age" can be defined, and that people who are experiencing old age form a recognizable group. However, there are many ways to define what "old age" actually means, and how it can be identified.

According to biology, ageing is a biological process by which a healthy, fit organism (for its environment) is converted to a less healthy and fit one. Some of the main ageing characteristics are:

- Reduced tissue/physiological function
- Increased susceptibility to disease (age-related diseases)
- Decreased resistance to stress.

Ageing can be understood as a biological process, as a stage within the life cycle, or in chronological terms. The former is a rather tricky definition to be used for the purposes of research, unless looking at specific groups (e.g., those experiencing a particular type of cognitive decline). Most frequently, however, researchers take the latter approach of using a chronological cut-off point to determine who is to be defined as old, which in some cases incorporates "older" people who are as young as

50. An alternative way of thinking about how old age can be defined is to explore how old people approach the issue. "Normal" ageing is ageing which occurs without disease. However, there are a number of physiological changes taking place that do not involve a pathological process. Although there may be bodily changes in the person, the person enjoys good function of mind and body and is able to live independently with a good quality of life. Usually, a distinction is being drawn between normal ageing (as a developmental process), which could comprise a number of indicators of physical decline such as arthritis, diabetes and blanks in memory, and real old age, which is signified by sharp declines in mental acuity (e.g. Alzheimer's disease) and social interactions (which often causes depression [1]) and other factors.

Because of the negative implications associated with aging, elders rarely perceive themselves in this way, preferring to see themselves as “getting on” rather than “getting past it”. Therefore, when defining what old age is, it is worth emphasizing that there are large numbers of people who, while identified as belonging to the “aged” fraction of our ageing societies, remain autonomous and do not consider themselves to be old.

Nevertheless, these individuals form a group that has certain characteristics in common. They are largely retired or semiretired, and are therefore likely to have undergone a notable life shift (in the move from employment). Further life changes may also have occurred or at least be on the horizon, including plans to downsize the family home or perhaps enter sheltered accommodation. Besides sharing various life circumstances, elders as a group may have certain attitudes towards the notion of family, friendship and their associated roles. These attitudes mark them out as a group with unique design requirements, and perhaps do so more strongly than the potential of their experiencing physical or cognitive decline.

1.2 Elderly People and Technology

The interplay between people and technology can be described as an ongoing dynamic process with two opposite sides. On the one hand, there are the positive-thinkers who support innovation and change which are seen as progress that can only be positive for humanity. On the other hand, there is the group of skeptical people with a critical

point of view that anything new has to be closely observed in order to be accepted and even then it must be handled with care. Both sides provide interesting points of view.

The main challenge when trying to introduce new types of technological and media inventions and innovations is to try to take advantage of the many new possibilities they offer while respecting and properly addressing the fears arising when introducing new forms of communication, education and living [2]. When it comes to elder people and their interplay with technology, two main aspects must be taken under consideration. The first one is acceptance. People have to accept new technologies first, in order to be able and willing to use them on a regular basis. The other critical aspect is creating an adaptive, ambient and inclusive design which will be discussed below.

Acceptance is a very essential point when studying the effect that new technologies have on the lives of elderly people. Younger people use technology in their everyday lives and are familiar with using and controlling technological devices. The elderly, on the other hand, are usually afraid to use technology. How can they be expected to use something that they are not willing to accept? Their lack of experience with technological products results into their reluctance in use. To address this problem the most frequent approach is to use familiar technological solutions such as TV-based services [3] in order to introduce new possibilities while minimizing the risks of rejection due to fears based on lack of experience. For example, in [4] a solution via IPTV is proposed that results into an affordable, unobtrusive, evolvable, usable and easily deployable platform which aims to approach the vision of “AAL for All”.

The second critical point to take under consideration, when designing new technological devices for the elderly, is the level of adaptation. In other words, this is the level to which the designed technological media can meet user requirements efficiently. Elder people are characterized by heterogeneity. This results in the existence of many user categories according to their cognitive abilities, speed of learning and curiosity or their degree of concentration. According to the user’s needs, designs made for the elderly must allow different usages. This can be accomplished

by using different micro-elements that react adaptively so that the information that the users gets is influenced by the pattern of usage as the device adapts to the user.

Moreover, when it comes to supporting elderly people in their everyday lives, it is more effective to use ambient media such as television and mobile phones. These appliances already exist in the elderly's everyday lives and are not considered to be disturbing, because people are already familiar with their use.

Finally, in order for a device to be universally acceptable it must conform to the principles of universal design. Design for all dictates that devices are produced for a large group of people and hence, heterogeneity should be taken into account. In order to achieve this, it is important to include the final users in the design and development process. As a result, elderly people need to be included in the testing and evaluation of the prototypes throughout the whole development process which produces a great impact on the acceptance of the final product.

1.3 Assistive Household Robotic Platforms

The field of assistive domestic robotic platforms for household use has been drawing considerable attention in recent years. As opposed to other assistive domestic robotics like automatic floor cleaners or pure surveillance robots, these social robotic platforms are designed to provide services to their human users through direct interaction, like displaying information, supporting communication with other people or simply entertaining the users [5].

The primary goal of these robots is to make their users feel safe and less lonely at home, while enabling and facilitating them in their independent or semi-independent lives at their own residences. In other words, they aim to advance towards a robot solution that will enhance wellness and quality of life for seniors, and enhance their ability to live independently for longer at their homes [6]. While technology and particularly innovative robots for ageing well at home can provide the means and solutions for helping the elderly achieve the desired independent longevity, elderly people's attitude, mistrust and suspiciousness towards anything new to them and especially anything that is hi-tech, is always a major setback.

Furthermore, employing the intelligence of the surrounding environment and adding it to the arsenal of robotic assistants, opens new promising horizons in the field of Ambient Assisted Living. The surrounding AmI environment can contribute significantly towards the augmentation of the context-awareness of robotic platforms which in turn can employ the offered possibilities of the intelligent environment towards building more complex behavioral scenarios. Furthermore, the assistive robots become more actively engaged with their users and the environment by exploring the functionality that they are offered which results into higher levels of acceptance and subjective satisfaction for their users.

Finding natural ways to interact with new technologies becomes a challenging field of research. Towards this objective, this work employs multimodal ways of interactions that provide a natural way of interfacing, aiming to increase the acceptance of such systems and increase their level of adaptation.

1.4 Scope and Objectives of this Work

The purpose of the proposed work is to provide a development framework that aims to facilitate developers in the process of designing and building interaction-friendly multimodal applications for elderly users. Taking into account the existing state of the art, the current work examines related research findings and takes a few steps further by providing developers with the necessary framework, tools and building blocks for creating interactive applications that are fit to the needs of the elderly users.

Furthermore, the developed framework guides the creation of applications that are designed and developed to be friendly to the elder users while introducing new interaction modalities such as voice and gestures, thus enriching and simplifying the interaction experience provided by the developed applications.

The modalities that have been implemented for the interaction process include voice, touch and gestures, as these modalities provide natural ways of interaction which are suitable for older people in terms of ease of use and subjective satisfaction, resulting in a smoother learning curve [7] [8]. Therefore, the framework provides instances of voice and gesture recognition engines to implement these approaches. All the controls in the proposed framework support activation through any of the aforementioned

modalities. The necessary tools to realize the pairing and matching between vocal commands, gestures and the user-interface scenes have be implemented as part of the framework, thus offering to developers a convenient workspace to build multimodal elderly-friendly applications.

1.5 Structure and Contents of this thesis

The remaining chapters of this report are organized as follows:

Chapter 2 provides an overview of the different research studies that have been carried out in the context of psychological aspects on the functioning of elderly people. Furthermore is goes through multimodal interaction techniques in ambient assisted living environments targeting the elderly and discusses the challenges that the elderly target user group imposes on the design. Moreover, it presents the aspects of human robot interaction in ambient intelligence environments and discusses the different adaptation options that that the multimodal interaction paradigm has to offer. Finally, it discusses existing technologies regarding the human robot and intelligent environment interaction as well as state of the art programming environments and tools in the same context.

Chapter 3 identifies the target user group that will be addressed in the current dissertation and its particular requirements. It identifies the problems encountered when designing software for household robotic platforms for the specified user group and discusses design methodologies that need to be applied when designing for this target group.

Chapter 4 presents the architecture of the proposed framework and the design methodology of the proposed solution, focusing on the framework modules that support the development of multimodal user interfaces that are elderly-friendly.

Chapter 5 discusses the implementation of the proposed framework, the extension of the ACTA scripting language runtime, the multimodal interaction context, the static and dynamic adaptation capabilities, the globalization and localization strategies, the state management of the different application dialogues and the augmented cognition capabilities that AmI environments can offer to domestic robotic platforms.

Chapter 6 addresses usability evaluation and testing and describes the methods that were used, issues specific to usability testing when the target user group is people of the third age, as well as heuristic evaluation case studies.

Chapter 7 presents use case applications developed using the FIRMA framework. Furthermore, it suggests case studies related to the proposed framework, adaptation mechanisms and supporting tools in applications made for domestic robots and the elderly in ambient assisted living environments.

Finally, chapter 8 focuses on the discussion of the proposed solution, highlighting the scientific contribution of this thesis and elaborates on ways that the proposed framework can be tweaked and extended in the context of future work.

2 Related and Background Work

This work aims at establishing the foundations for facilitating the design and development of elderly-friendly multimodal interactive applications targeted to household robotic platforms which are deployed in Ambient Intelligence Environments. This discusses the state of the art in the targeted application domains.

2.1 Psychosocial aspects on the functioning of elderly people

As mentioned in chapter 1, the percentage of the people in the oldest age group (more than 75 years old) is rapidly increasing worldwide. The growing need for providing medical, physical and psychological care for this vast heterogenic group is in the focus of attention of most highly developed countries due to the excessive costs of support. But not all of the elderly residents need such intense help. The deterioration in the cognitive and physical functioning connected with getting old can be split in three major categories [9], [10], [11]:

1. Successful ageing
2. Normal ageing
3. Diseased ageing

The successful ageing group (8-10% of the general population of the elderly people) stays active and independent mostly until they become the so called oldest old people (older than 85 years old), and even then they require relatively little support, mostly connected with carrying heavy objects. For this group, having an assistive robot would be a chance for staying as independent and active (physically, socially and cognitively) as when they were much younger.

The normal aging group, which covers most of the general population of the elderly people, suffer from typical age-related problems such as: somatic complains due to different medical conditions, age-related cognitive decline and senses deterioration. This group needs moderate support, mostly provided by the closest members of the

family and General Practitioners. It is estimated that almost 70% of the elderly people need some kind of aid such as walking sticks, glasses or hearing aids [12], [13]. Just like the successful aging group, people in this group would welcome help provided by an assistive robot. The nature of the required help would be very similar but it should be provided earlier in time.

Both successful and normal aging group have similar needs for physical and emotional support which increase year after year. Most of the elderly people are able to look after themselves but they require support in more demanding household jobs. They are able to fulfil most of their needs by themselves, such as keeping in touch with social contacts and receiving emotional support gained from their relatives and friends.

On the other hand, the diseased ageing group requires larger amounts of support due to its profound physical or cognitive issues. This group consists of the people suffering from cognitive impairments as well as from other diseases with physical manifestations which make them unable to function independently. It is estimated that a high percentage of the people of this group suffer from more than one serious medical condition. Their everyday functioning is highly influenced by their health issues and they require physical help much earlier than the groups mentioned above. The use of an assistive robot would be of invaluable help not only for the elderly people of this group, but for their families as well.

One important challenge in Human-Robot Interaction is the relatively low level of technology acceptance among the elderly people. It is a fast growing field of research but so far the gathered data is not conclusive. Most of the researchers underline the existing correlation between the age, general technical skills, familiarity with technology and the social influence on Human–Robot Interaction acceptance [14]. The oldest generation currently researched is not familiar with the sophisticated IT solutions and presents a higher fear of technology. However, taking into consideration the fact that it will take quite some time for assistive robots to become a mature scientific field and reach its full potential, the real target group for the HRI is currently in their 50s and use technology much more often than these target user groups did 20-30 years ago.

The problems frequently highlighted in the elderly population group could be split into these four major subcategories:

- Cognitive impairments
- Emotional and social withdrawal
- Everyday functioning problems
- Senses deterioration

Even a slight deterioration in one subcategory influences the other ones and reduces the quality of life of the elderly person. Therefore the provided help should not only target the mostly impaired subcategory but also focus on maintaining the relatively preserved ones on a constant level.

Age associated memory impairments (AAMI), which are mostly subjective complains on the decline of cognitive functions, are vastly observed in the elderly user group (up to 50% in the "60 - 69 years old" group, but already 85% in the "more than 85 years old" group [15], [16], [17], [18]). Some of the AAMI cases are followed by brain tissue deterioration that can be observed in neuroimaging, but on the most part they don't show up in the results of psychological tests. These people report cognitive impairments which can't be detected by standard neuropsychological tools although such impairments are present in their everyday functioning. This is called "The Einstein effect" and it has been described in the high functioning group due to these people's very efficient compensation mechanisms [19].

Emotional and social withdrawal is mostly connected with the gradual decreasing mobility capabilities of the person as well as sociological changes in the structure of our society. The multigeneration type of family is nowadays replaced by a nuclear family pattern which inevitably creates a generation gap and reduces the number of close relatives. The economy of the developed countries has forced their residents to travel for work which in turn reduced the already limited amount of social interaction between generations. The elder generation tries to overcome such void with peer interaction but as the time goes by, the increased amount of deaths among their closest friends, results into increased feelings of social exclusion. Although nowadays'

enhanced technological advancements provide the possibility for communication with the remaining family members regardless of their location, elderly people hardly ever use such channels of communication due to low self-esteem and restricted learning mechanisms.

Everyday functioning problems occur due to the decreased physical capability and executive function impairments of the elderly. Within the normal and successful ageing group only the physical inabilities have an impact on everyday functioning. Elderly people are usually coping with housework and everyday problems by slowing the pace of their actions. Providing them with a little help such as meals-on-wheels, incoming nurses or caregivers is adequate to maintain the previous independent functioning.

The authors of the recent World Alzheimer's Report 2015 [20] underline that within European countries, the differences in the care provided to the elderly residents are profound. The increasing and dominating institutionalized type of care in the Western Europe is in opposition to the home care provided in Eastern Europe. These two distinctly different patterns of care are the result of the economic differences between regions as well as the demographic changes and particularly the decrease of the birth rate.

This differential pattern of care provision creates a two-speed society. While on one hand, the institutionalized help pattern provides inherent social contact with a wide group of people, on the other hand there is little contact with individual family members. The level of needed medical care and professional therapy is higher while the need for performing everyday activities such as cooking, cleaning and shopping decreases.

The exact opposite pattern is observed in the Eastern Europe, where the non-institutionalized pattern of care is dominating. Elderly people stay either with their children or in their own apartments but they are looked after by their children if and when necessary. However, in both cases, the main unfulfilled need is staying independent while not feeling (to be) a burden to their families (economically, emotionally and physically).

It is estimated that almost 70% of the elderly people need different kinds of aids due to the deterioration of their sensory organs [13], [21], [22]. They majority suffers from disturbed eyesight due to different medical conditions such as cataract and short or long sightedness. The second most often deterioration is deafness. They both have significant impact on the person's everyday functioning, especially on the quality of the person's social interactions. For example, it is hard for younger people to imagine the depths of these impairments and they do not adjust their voice to the needs of the elder generation. Not only the volume but also the pitch, the pace and the clarity of speech is crucial for the correct understanding of the spoken language. Most of the elderly people withdraw from the social contacts due to fear of inadequate reactions, as well as fear of being mocked, fooled or used.

2.2 Designing for the elderly

Several user studies showed that elderly people and their families regard social inclusion, safety and home automation as important features of future homecare environments. With respect to interaction in such environments, one of the main research challenges is the design of adequate user interfaces. This is due to the fact that elderly people vary considerably in their physical and cognitive abilities, which makes it difficult to use traditional interaction models [23]. Focusing on single interaction strategies may not always provide appropriate solutions [24], as many older computer users are affected by multiple disabilities, and such multiple minor (and sometimes major) impairments can interact.

To address this problem various authors developed intelligent user interfaces, which support users according to their individual needs. For example, [25] introduces a spatial metaphor for universal control devices to structure available services based on the elderly person's own apartment. The results of the study showed that the Apartment Metaphor is actually appropriate to enable elderly people to access a large number of services available in an AAL environment in an intuitive way. The metaphor showed a way for structuring and visualizing services in a universal control device.

Furthermore, [26] presented a novel general framework for multimodal dialogue processing, which is conceived following an application-independent philosophy. In

fact, it is able to manage multimodal communication between people and the environment in different application scenarios. The core of framework architecture is composed of the analysis and planning levels, which enable the processing of information derived from whatever input modalities, giving these inputs an appropriate representation and integrating these individual representations into a joint semantic interpretation.

Finally, [27] presents a prototype for a Web 2.0–enabled ambient assisted living (AAL) device that offers easy-to-use functionality to help elderly people keep and establish new contacts, find events that match their interests and be aided in sustaining their mobility. The prototype consists of a hardware device for mobile usage feasible to host the desired functionality while being adequate for the use by elderly people. An internet tablet was selected accommodating a large touch screen. The study focuses on analyzing all possible transportation routes of a local city or area and providing to the elders the most convenient way of transportation for their desired destination. Based on either an event suggestion or a contact person, the users of the device could have a travel advisory calculated to reach the event or person. For routing, any applicable means of transportation were taken into consideration. For example, taxi services that do not offer handicapped accessible vehicles were excluded for a person who used a wheelchair. Also, the maximum distance a person can walk could be specified.

Intelligent user interfaces can have many different manifestations. The field of Ambient Intelligence (AmI) has contributed towards the utilization of different modalities and interaction metaphors in order to achieve natural interaction and increase the level of acceptance of modern technology by the older users. The emergence of Ambient Intelligence is leading to the elaboration of new interaction concepts that extend beyond current user interfaces based on the desktop metaphor and menu driven interfaces, thus driving a transition to more natural and intuitive interaction with everyday things [28]. Natural interaction refers to people interacting with technology as they are used to interact with the real world in everyday life, through gestures, expressions, movements, etc., and discovering the world by looking around and manipulating physical objects [29]. Typical examples are input techniques

such as touch, gestures, head and body position tracking and manipulation of physical objects, which seamlessly integrate the physical and digital worlds and support the direct engagement of the user with the environment [30]. Augmented Reality (AR) allows virtual imagery to augment and enhance physical objects in real time. Users may interact with the virtual images using real objects in a seamless way [31]. Progress in computer vision approaches largely contributes to innovative interaction in AmI environments through techniques such as like image acquisition, image processing, object recognition (2D and 3D), scene analysis, and image flow analysis, which can be exploited for humans' and objects' recognition and tracking [32]. At the same time, ICT components are embedded into everyday objects like furniture, clothing, white goods, toys, etc. [33]. Augmented objects can be used for providing implicit or explicit input to systems while their physical and mental existence as computational devices disappears [34]. Ambient interaction merges real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exist and interact in real time. Additionally, in Ambient Intelligence environments interaction is monitored and implicit input is also extended to include empathy to understand human's feeling or states.

2.2.1 The elderly and Touch-based interaction

The elderly user group is a very special user group due to the heterogeneity that appears among its users. Old age takes its toll on people, while introducing numerous types of impairments and minor or major disabilities. Thus, researchers have focused on finding techniques to facilitate the elderly when interacting with touch or gesture – based interactive applications. These research findings support the preliminary hypotheses one would make about the elderly.

Nicolau et al. [35] while performing research on elderly text-entry performance on touch screens found that an optimal inter-key threshold must be defined and key presses inside this timeframe should be considered to be insertion errors and should be discarded. This inter-key threshold varies in respect to the device that is being used. The research showed that the optimal inter-key threshold should be about 100ms when using mobile phones and 150ms when using tablets. Moreover, tablet devices were found to compensate some of the challenges imposed by mobile devices, either

due to larger key sizes and/or due to static positioning. Significant decrease of both insertion (1.7%) and substitution (4%) errors were observed when tablets were used. Furthermore, the authors suggest that the elderly participants theoretically benefit from a layout shift in the bottom-right direction as most substitution errors occur in this direction and whenever possible keys should be wider instead of taller as width is a more important factor that reduces errors. On the other hand, the spacebar was suggested to be narrower, extending from the middle of key C to the middle of key B. Finally, dealing with poor aiming was found to be more important than compensating for finger slips and the use of language-based correctors was found to be beneficial as cognitive errors were quite common among elderly users. Simple language-based solutions were proven to provide a suitable answer to these types of errors (for example, to deal with blank space omissions or substitution of similar letters such as p→q, m→w).

Kobayashi et al. [36] studied the interactions between the elderly and mobile touch screens. Research showed that larger targets (8 mm or larger in size) should be used while the gap between intended and actual touch locations should be addressed. Moreover, the use of drag and pinch gestures should be considered rather than simple taps and focus should be given to explicitly displaying the current mode of the applications as the elderly seemed to become confused otherwise overtime.

Hollinworth et al. [37] investigated familiar interactions to help older adults learn computer applications more easily. Research findings showed that familiar visual objects should be used for interaction as well as familiar behaviors should be attributed to these objects. Seldom used application functionality should be removed while avoiding incorporating hidden functions. Familiar interfaces were found to help in orienting older users with an application, and that functionality could be made more explicit by exploiting prior knowledge and skills of the real world that people already possess. Finally, actions in many cases should be made much simpler and easily reversible, which can help to encourage novice users to explore what the applications have to offer.

Wacharamanothan et al. [38] evaluated a new touchscreen input method for elderly users with tremor, called swabbing. Swabbing is a selection method for touchscreens

where all the selectable targets are placed on the circumference of a circle that is inscribed in the desired selection area. The users are able to select a target by placing their fingers in the center of the circle and then slowly moving towards the desired target. The corresponding target that lies on the extension of the user's finger trajectory is calculated and then highlighted. The user can select any highlighted target by lifting his finger before or even after having crossed the desired target. This "sliding" technique was shown to reduce intentional tremor. The researchers found that tapping is a viable choice for square targets that are at least 54 mm wide. When the target width is smaller than 41 mm, swabbing becomes a better alternative. Thus, elderly users with tremor may prefer a more accurate input method, even if it is slower.

Stößel et al. [39] focused on mobile device interaction gestures for older users. They found that gestures which are suitable for older users might be different from the ones that are designed with younger users in mind. The researchers stated that if we want to exploit the potential benefits that gesture-based interaction could bring about, we need to carefully design interaction patterns suited to older people's needs and abilities.

Leonardi et al. [40] made an exploratory study of a touch-based gestural interface for the elderly. They found that tap gestures (when applied to well-recognized objects) are the easiest ones to understand and remember. However, the definition of the tap (for example, how much time is allowed between touch and release) should be carefully considered and possibly automatically adapted. Moreover, their research showed that tapping on the background outside of an object to perform some actions on that very object sounded unintuitive and should be avoided because the idea of "tapping on nothing" (that is, on the background) is very difficult to communicate. Furthermore, the authors suggest that objects should not be overloaded with actions performed both by a tap and by a drag gesture because in case of insufficient pressure or in case of false starts the two gestures may be easily confused. To this extent, for drag gestures, a "natural" version should be implemented: when the touch is lost during a drag the object should stay where it has been left rather than flying back to its initial position. Iconic gestures (gestures that visually and analogically represent

their meaning, e.g., drawing a rectangle to fire the "create new message" function) were found to be very engaging and their hedonic value should not be underestimated as a way to motivate users with high computer anxiety. On the other hand, a proper setting of the time parameters of gestures is of paramount importance. In this respect, the possibility of automatic adaptation by the system should be seriously considered because of the large variability in touch performance by elderly people (due to age, health-related issue, etc.). Finally, animations alone were not found to be effective in signaling synchronous and asynchronous events on the interfaces and research findings suggests that they should be accompanied by redundant information in other modalities.

Finally, Jako et al. [41] investigated the effects of multimodal feedback on the performance of older adults with normal and impaired vision. Their research showed that non-visual feedback and additional feedback (in addition to visual) have the potential to either maintain or improve task performance while the effectiveness of non-visual feedback (and multimodal feedback generally) is more apparent as visual acuity loss becomes more severe.

2.2.2 Speech Recognition and the elderly

Input through speech is not particularly new. Research has been conducted in this field for years, and today thousands of commercial and research product have been developed. Although much work has been done to date, the research field of human language processing is more than active mainly due to the fact that modern speech interfaces enable the communication through identification of simple voice commands on a predefined grammar [42]. In the context of Ambient Intelligence, although desirable, human language understanding is not crucial. However, it is crucial to provide the means for error free detection of speech when such input is required [43]. This is essential mainly because Ambient Intelligence aims at making the interaction with computing technology transparent to the user from the one hand and as affective as the usage of traditional means. It is therefore argued that the adoption of new technologies resides at least on being as affective as the main stream technologies used at the time of their presence [44].

Furthermore when designing speech recognition systems for the elderly target user group, its heterogeneity should be taken into account. Individual persons have different individual needs based on their different age related health impairments. Such impairments can affect the person's speech capabilities which results either in an increasingly limited vocabulary or fluctuations in their pronunciation clarity [45]. These limitations should be taken under consideration when designing the parameterization properties of such speech engines.

2.2.3 Gesture Recognition and the elderly

Gestures are employed in the context of ambient intelligence for providing alternative ways of user input in a human like fashion. Gestures when used in the context of human to human communication are a quick and intuitive way of communication. On the contrary, identifying human gestures in a computerized environment is not an easy task. Research conducted in this field involves the usage of gestures for providing input to augmented desk interface systems using multiple fingertips recognition (identify fingertips and their trajectories and infer gestures based on these trajectories) [46], [47]. In the same context, computer vision is used for identifying hand gestures, facial expressions and body postures [48]. Furthermore, the usage of thimble-shaped fingertip markers made of white printing paper with a 'black light' source has been proposed for providing gesture recognition in the context of back projection walls [49]. Finally, gaze recognition has been employed for facilitating alternative gesture based input [50], [51], [52].

Furthermore the heterogeneity of the elderly user group implies various restrictions when designing and developing gestural interaction modalities for the elder users. Individual persons have different individual needs based on their different age related health impairments. Such impairments can affect the person's mobility capabilities and cognitive functions which results in mobility restrictions in different body parts and difficulties in remembering the specified gestures and the optimal way of performing them. These limitations should be taken under consideration when designing the parameterization properties of such gesture recognition engines.

2.3 Design for All

The term "design for all" or "universal design" describes the concept of designing environments, products and services to be accessible and usable to the greatest extent possible by everyone, regardless of their age, abilities, disabilities or status in life. The elderly user group in particular, is a very heterogeneous user group due to the different impairments, limitations and disabilities that the elderly have. As a result, when designing environments, products and services for the elderly the seven principles of universal design [53] should be taken under consideration:

- **Equitable Use.** Equitable use refers to the fact that the design should be marketable to people with diverse abilities. In order to achieve this we should make sure to provide the same means of use for all users. Identical whenever possible; equivalent when not. We should also avoid isolating or stigmatizing any users. Provision for privacy, security, and safety should be made equally available to all users. Finally the design should be made appealing to all.
- **Flexibility in Use.** Flexibility in use implies that the design should accommodate a wide range of individual preferences and abilities. The users should be provided with choices in methods of use. Both right and left access and use must be accommodated. User's accuracy and precision should be facilitated and adaptability to the user's pace should be provided.
- **Simple and Intuitive Use.** This principle implies that the use of the design should be easy to understand, regardless of the user's experience, knowledge, language skills, or current concentration level. Thus unnecessary complexity must be eliminated. The design should be consistent with user expectations and intuition and a wide range of literacy and language skills must be accommodated. The information should be arranged consistently with its importance and effective and prompting feedback must be provided during and after task completion.
- **Perceptible Information.** Perceptible information means that the design communicates necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities. In order to achieve that the use of different modes (pictorial, tactile, verbal etc.) for redundant

presentation of essential information is mandatory. Furthermore the design must maximize the "legibility" of essential information and differentiate elements in ways that can be described such as it will be made easy to give instructions or directions. Finally compatibility with a variety of techniques or devices used by people with sensory limitations must be provided.

- **Tolerance for Error.** In order for the design to be error tolerant, it should minimize hazards and adverse consequences of accidental or unintended actions. The design elements should be arranged in such a way that hazards and errors are minimized which means that we should make accessible the most used elements while isolating or shielding the others. Appropriate warnings of hazards and errors must be provided along with fail safe features in order to discourage unconscious action in tasks that require the undistracted attention of the user.
- **Low Physical Effort.** The users should be able to use the design efficiently and comfortably with a minimum of fatigue. To accomplish the low physical effort principle, the user must be allowed to maintain a neutral body position and use reasonable operating forces while minimizing repetitive actions and sustained physical effort.
- **Size and Space for Approach and Use.** The design should provide the appropriate size and space for approach, reach, manipulation and use regardless of user's body size, posture or mobility. In order to achieve this, there should be provided a clear line of sight to important elements for any seated or standing user. Moreover, the reach to all components should be made comfortable. Variations in hand and grip size should be accommodated appropriately while providing adequate space for the use of assistive devices or personal assistance.

In the context of HCI, Design for All implies an explicit design focus on systematically addressing diversity, as opposed to afterthoughts or ad hoc approaches [54]. In the emerging Information Society, therefore, Universal Access becomes predominantly an issue of design, and the question arises of how it is possible to design systems that take into account diversity and satisfy the variety of implied requirements. Research

work in the past two decades has highlighted a shift of perspective and reinterpretation of HCI design, in the context of Universal access, from current artifact-oriented practices towards a deeper and multidisciplinary understanding of the diverse factors shaping interaction with technology, such as users' characteristics and requirements and contexts of use, and has proposed methods and techniques that enable to proactively take into account and appropriately address diversity in the design of interactive artifacts [54]. In the framework of such efforts, the concept of design for all has been reinterpreted and redefined in the domain of HCI. One of the main concepts proposed in such a context as a solution for catering for the needs and requirements of a diverse user population in a variety of context of use is that of automatic user interface adaptation [55].

2.3.1 UI adaption for the elderly

Research efforts in the past two decades have elaborated comprehensive and systematic approaches to UI adaptations in the context of Universal Access and Design for All [56]. The Unified User Interfaces methodology was conceived and applied [54] as a vehicle to efficiently and effectively ensure, through an adaptation-based approach, the accessibility and usability of UIs to users with diverse characteristics, supporting also technological platform independence, metaphor independence, and user-profile independence. In such a context, automatic UI adaptation seeks to minimize the need for a posteriori adaptations and deliver products that can be adapted for use by the widest possible end-user population (adaptable UIs). This implies the provision of alternative interface instances depending on the abilities, requirements, and preferences of the target user groups, as well as the characteristics of the context of use (e.g., technological platform, physical environment). The main objective is to ensure that end-users are provided with the most appropriate UI instance at runtime, based both on group or individual characteristics.

As the elderly are a highly heterogeneous target user group, user interface adaptation is crucial to ensure accessibility, usability and acceptance of interactive technologies for the aging population. Various research efforts have investigated adaptable and adaptive user interfaces for the elderly.

Leuteritz et al. in [57] present an approach towards ensuring high quality interaction for older users, building on personalisation and adaptation techniques. The purpose is to facilitate the development of interactive applications and services for different platforms, to develop various accessibility components that can be used across a range of interaction devices, and to enable the personalisation of interaction, as well as automatic tailoring-to-device capabilities and characteristics, thus offering an individualised user experience. Leonidis et al. [58] proposes an adaptation development toolkit and related widget library, which directly embeds lexical level adaptations into common interactive widgets. The toolkit has been designed for the development of adaptable applications for older users. An adaptable card game for older users is described in [59]. The game cam modify its user interface, difficulty level and playing mode according to the users' profile.

A pattern-based approach to web forms adaptation for the elderly is presented in [60]. These patterns allow for dynamic substitution and/or augmentation of user interface parts at runtime, with the goal of improving the individual usability for an elderly user in a specific use context. This approach could eventually lead to highly personalized web forms. A pattern-based approach is also adopted in [61], where usability problems which may be caused by adaptations are investigated, including disorientation and the feeling of losing control. Design strategies to overcome the drawbacks of adaptations are proposed to increase the transparency and controllability of run time adaptations. An experimental user study to investigate the effectiveness and acceptability of the proposed patterns in different cost-benefit situations and for different users is presented. The patterns turn out to increase the transparency and controllability of adaptations during the interaction. They help users to optimize the subjective utility of the system's adaptation behavior. Moreover, the results suggest that preference and acceptance of the different patterns depend on the cost-benefit condition. Finally, [62] elaborates user requirements for UI adaptation of the elderly and stroke survivors. User requirements are defined for general, physical/motor, sensory, cognitive, and environmental aspects. The user characteristics found to be of most relevance for adaptive user interfaces are:

- Physical/motor: personal mobility, hand strength, dexterity and finger movement, muscular control including semi-paralysis and speech impairment. Sensory: vision – diverse impairments, touch sensitivity and hearing impairments.
- Cognitive: memory impairments (long-term and working memory), reaction time, thinking speed, literacy (including computer literacy), learning ability, concentration, language comprehension and motivation.

A number of environmental characteristics are found to be important also: lighting (adequacy, quality, controllability, glare, flickering), ambient noise, physical layout of environment – devices – TV, remote, sensors, etc., and layout of controls – taking into account cognitive, physical/motor and sensory limitations.

2.4 Human Robot Interaction

Human–robot interaction (HRI) is the study of interactions between humans and robots. HRI is a relatively young discipline that has attracted the interest of many researchers over the last years due to the increasing development of complex robotic platforms and people's affordance to such pieces of technology in their everyday lives. Furthermore, robots are being used more and more in real world applications such as eldercare, education, therapy or rehabilitation etc [63].

2.4.1 Interaction categories

Human–Robot Interaction (HRI) is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans. Interaction between a human and a robot may require the use of different modalities and may take several forms, but these forms are largely influenced by whether the human and the robot are in close proximity to each other or not [64]. Hence, interaction can be separated into two general categories:

- The first category is remote interaction, where the human and the robot are not located in close proximity or are separated spatially or even temporally (for example, the Mars Rovers are separated from earth both in space and time).

- The second category is proximate interaction where the humans and the robots are in close proximity with each other, for example service robots usually are in the same room or house as their users.

The category in which the respective human-robot interaction falls can have different implications on the quality of the interaction as well as limitations on the used interaction modalities. For example, controlling a robot at a distance (in a remote interaction scenario) using a fixed point depth camera has lower complexity and higher accuracy than trying to implement the same interaction modality using a sensor onboard the robot. In the latter approach, different factors have to be taken in account, such as the distance between the operator and the robot, the relative position between them and possibly other dynamically changing environmental factors such as the relative position of the robot in its surroundings etc.

2.4.2 Interaction modalities

Dealing with uncertainty in sensing and real-time perception are some of the most tricky and ongoing challenges of robotics. HRI faces particularly complex perceptual challenges, because the perception, understanding, and reaction to human activity is required in real-time. The range of sensor inputs for human interaction is far larger than for most other robotic domains in use today. HRI inputs include vision and speech, both major open challenges for real-time data processing. Computer vision methods that can process human-oriented data such as facial expression [65] and gestures [66] must be capable of handling a vast range of possible inputs and situations. Similarly, language understanding and dialog systems between human users and robots remain an open research challenge [67], [68]. Tougher still is to obtain understanding of the connection between visual and linguistic data [69] and combining them toward improved sensing [70] and expression [71]. Even in the cases where the range of input for HRI-specific sensors is manageable, there is the added challenge of developing systems that can accomplish the sensory processing needed in a low-latency timeframe that is suitable for human interaction. For example, Kismet [72], an animated robotic head designed for infant-like interactions with a human, using object tracking for active vision, speech and prosody detection and imitation, and an actuated face for facial expressions, required several computers running in

parallel to produce engaging facial and speech behavior. The humanoid ASIMO has been adapted to use a combination visual-auditory system for operation in indoor environments [73]. ASIMO's subsystems were used for perception, planning, and action with the goal of enabling human-robot interaction. Adding meaning to the facial and physical expressions and speech, and combining all of those capabilities in real time on a mobile, self-contained robot platform, is still an open research problem in HRI. Even though most implemented HRI systems are necessarily domains-specific, as all physical systems, they still require the additional step of generalization to make them work beyond the research lab context. Computer vision solutions often depend on specific lighting conditions [74], ambient colors [75], and objects in the scene [76]. Beyond the lab, either the environment must be constrained to match the acceptable conditions for system operation [77], or the system capabilities must be extended in order to meet the range of conditions in the specific destination environment [78].

In addition to robot sensors that mimic the functionality of human perception (speech recognition, computer vision, etc.), sensors are being developed that cater for the unique perceptual capabilities of robots. These sensors enable a machine to observe people and the environment in ways that may be beyond human access. Physiological signals, such as heart rate, blood pressure, galvanic skin response (GSR, the measure of skin conductance using a galvanometer), provide information about the user's emotional state [79], [80], [81] that may not otherwise be observable. Work by Mower et al. [82] used GSR as part of an HRI system to model and predict when a user is about to quit a rehabilitation-type task. Body pose and movement are important sources of information for social interaction [69]. For example, social and expressive gestures are crucial components of human-human and human-robot interaction [83]. Computer vision can provide such information in limited contexts. In others, wearable sensors may be an effective means of obtaining human activity data in real time with high accuracy [84]. Such wearable systems have been used in HRI tasks applied to physical rehabilitation post-stroke [85], and for social interaction [86]. In addition to developing new and improving existing sensors toward particular needs of HRI, researchers are also developing algorithms for integrating multi-sensor multi-modal data inherent to HRI domains [87], [88], [89], [90], [73]. For example, Kapoor and

Picard [91] implemented an affect recognition system that applies Gaussian models to fuse multiple sensors. Multi-modal sensing has also been used for a robot to detect the attention of human users in order to determine if a user is addressing the robot [92], integrating person tracking, face recognition [93], sound source localization [94], and leg detection [95].

2.5 Introducing Human robot interaction in AmI environments

2.5.1 Ambient Intelligence (AmI)

Although serious efforts have been made towards providing the citizens of the emerging Information Society with seamless access to a large number of computer applications, these attempts mainly focus on adapting traditional interaction techniques to the context of use and the specific needs of the corresponding users. On the other hand, Ambient Intelligence (AmI) promotes the vision that "smart" environments should be able to react in an attentive, responsive, active and adaptive way to the state and the actions of human beings and objects, providing them with the appropriate services. Ambient Intelligence is an emerging field of research and development that has managed to capture the attention of researchers and developers, especially in Europe [96]. Ambient Intelligence is introducing a shift from traditional interaction techniques to new, novel concepts of communication with computer systems. AmI targets to distribute, embed, coordinate and interactively deliver computing intelligence within the surrounding environment. AmI technologies integrate sensing capabilities, processing power, reasoning mechanisms, networking facilities, applications and services, digital content, and actuating capabilities distributed in the surrounding environment. AmI will have profound consequences on the type, content and functionality of the emerging products and services, as well as on the way people will interact with them, bringing about multiple new requirements for the development of the Information Society. While a wide variety of different technologies is involved, the goal of AmI is to either hide the presence of technology from users, or to smoothly integrate it within the surrounding context as enhanced environment artifacts. This way, the computing-oriented connotation of technology

essentially fades-out or disappears in the environment, providing seamless and unobtrusive interaction paradigms. Therefore, people and their social situation, ranging from individuals to groups, and their corresponding environments (office buildings, homes, public spaces, etc.), are at the center of the design considerations [96].

AmI environments are designed to be unobtrusive, interconnected, adaptable, dynamic, embedded, and last but not least intelligent. In these environments, the old fashioned computer input and output devices disappear and are subtly substituted by other means of interaction. Instead, processors and sensors are integrated in everyday objects and tangible artifacts. So, for example, instead of using keyboards and computer screens, users are given the possibility and the opportunity to communicate directly with their clothes, household devices, and furniture. Moreover, entities may be able to communicate with each other and with other people's devices and furniture. The envisioned AmI environment is sensitive and dynamically aware of the needs of its inhabitants, and capable of anticipating their needs and behaviors. It is continuously aware of their personal requirements and preferences context-wise, and interacts with people in a user-friendly way, even capable of expressing, recognizing and responding to emotion [97].

The AmI environment is constituted of embedded and embodied low-cost hardware, providing advanced networks of heterogeneous information appliances or smart tangible artifacts. These artifacts can assist users in their everyday activities. Moreover, they can demonstrate common sense and problem-solving ability either individually or as a whole, and assume responsibility or take action for certain needs of the users such as monitoring, alerting and helping them in their tasks. Advances in AmI are facilitated or sometimes even enabled for the first time by the long-term increases in the power of microprocessors, their continuous struggle for a declining pattern in their power consumption and nanotechnology, as well as the cost-efficiency of storage capacities and communication bandwidths.

One domain application of ambient intelligence that has attracted much attention is support for independent or semi-independent living for the elderly. As human life expectancy has increased through improvements in public health and lifestyle, more

and more people live long enough to age and in some cases to suffer from ageing-related problems. In this context, the application of ambient intelligence is anticipated to be very promising.

2.5.2 Ambient Assisted Living

Smart Environments also called Smart Homes or Smart Houses are often topics of research projects such as, e.g., GENIO [98], MavHome [99], iDorm [100], etc. People with disabilities in particular could benefit most from such environments, as they offer basic assistive functionalities enabling increased independence and personal freedom and hence resulting in improved quality of life.

Facilitating the elderly in Smart Environments often requires the participation of other individuals such as relatives, other members of the family or even dedicated care givers.

An assisted living residence or assisted living facility (ALF) is a housing facility for people with disabilities. These facilities provide assistance supervision for people, normally seniors, for whom independent living is no longer appropriate, but who do not need the 24-hour medical care provided by a nursing home. Assisted living is a philosophy of care and services promoting independence and dignity [101]. Assistance may include the administration or supervision of medication, or personal care services provided by trained staff.

The term "Ambient Assisted Living" (AAL) refers to the meaningful usage of information and communication technologies so as to improve the quality of life of older people by giving them safety and comfort to stay longer at home [102]. The demographic changes over the last decades concerning the unequal growth of the population of people crossing into the third age territory have pointed out the need for a turn to Ambient Intelligence (AmI) concepts and technologies to support people in maintaining their well-being. These AmI concepts and technologies should address user needs by focusing on the safety and protection of the personal environment, while stimulating and enabling elderly people to maintain an active lifestyle.

Statistics from the World Health Organization indicate that the percentage of people who fall each year increases accordingly to their age. About a third of those who fall suffer injuries that reduce mobility and independence and increase the risk of premature death. In addition, depression, fear of falling and other psychological problems are common consequences of repeated falls. Within this context, it is clear that there is an important role for intelligent environments to provide a feeling of safety and protection by means of continuous contextual monitoring and immediate response in case of events with high health risks.

The process of normal aging inevitably introduces several forms of physical decline and cognitive impairment. Such functional limitations can lead to situations of reduced mobility and in turn to reduced social interaction. AmI environments can improve the feeling of self-efficacy by offering solutions for stimulating physical and cognitive fitness and enabling active participation in society.

Improving the quality of life for physically or cognitively impaired people and the constantly increasing proportion of the elderly is becoming a more and more essential task for today's European societies. From the perspective of an inhabitant, it is essential to keep in mind that residents consider their home to be a comfortable and safe place to live in. Technology is often considered to be an intruder in this environment [103]. In particular, the elderly may be afraid of technology mostly because it makes them feel that they are losing control over their house. Some people even fear the use of technology in their home. So the terms "technology" and "home environment" are not always a perfect fit [104]. When it comes to ambient assisted technology for the elderly and impaired people, another aspect must be taken under consideration. Many solutions are proposed to support people in everyday activities, but it is also important to provide the users with solutions that increase their participation in their everyday life. For example, instead of a digital "majordomo" who acts on behalf of the owner and takes care of everything automatically, it is equally important and sometimes preferable to provide the users with a digital assistant who tries to help them achieve the desired results and fulfill their everyday tasks more effectively. The goal of equipping the home environment with technology isn't just to automate all the tasks that are usually carried out by the residents. The main objective

in design is to provide tools and services that empower and enable people themselves to address their social, rational, and emotional needs. Equality, autonomy, and control are the goals of empowering design.

An ambient assisted living environment can provide a constantly monitored space, thus resulting into a safer householder (activity monitoring), automate specific tasks that a householder is unable to perform on his own (e.g., turning electrical appliances on or off) and provide a safer and potentially more secure environment by alerting the resident of potentially dangerous activities. Furthermore, it can alert helpers or care givers should the inhabitant be in a potentially dangerous, hazardous or life threatening situation (e.g., through linking to a local community alarm scheme). The aim and primary goal is to enable and empower the user, as well as support the independent living of householders (by giving prompts that can be auditory and/or visual).

2.6 Enhanced Human robot interaction through Adaptation

2.6.1 Adaptation based on modality selection

In today's pursuit of more transparent, flexible, and efficient human-computer interaction, a growing interest in multimodal interface design has emerged [105]. The goals are twofold: to achieve an interaction closer to natural human-human communication, and to increase the robustness of the interaction by using redundant or complementary information.

Multimodal systems are radically different from standard GUIs, largely because of the nature of human communication, and their basic architecture reflects these differences. Whereas input to GUIs is atomic and certain, machine perception of human input such as speech and gesture is uncertain, so any recognition-based system's interpretations are probabilistic. This means that what were formerly basic events in a GUI, such as object selection, now are events that require recognition and are subject to misinterpretation. Secondly, whereas standard GUIs assume a sequence of discrete events, such as keyboard and mouse clicks, multimodal systems must

process two or more continuous input streams that frequently are delivered simultaneously [106].

Multimodal systems have been shown to be more efficient, reliable and robust than traditional systems, e.g., [107], [108], [109], [110]. However, there are also contradictory findings: For instance, it has been shown that although multimodality may lead to higher perceived quality, users often do not use multimodal interaction strategies and instead stick to one input modality [111], [112], [113]. A possible explanation for these ambiguous results may be found in the fundamental differences of the systems used in these studies: the majority of experiments supporting the advantages of multimodal interaction where either focusing on spatial-verbal tasks investigating parallel speech-pen resp. speech-gesture input or on the effects of multimodal and/or multimedia output [109], [110].

Based on the aforementioned findings this research work focuses not just on providing alternative modalities for the users and thus relying to the user to select which modality to use, but rather integrates logic to the system making it capable of selecting at run-time the most appropriate modality based on the profile of the user and the environmental situation.

2.6.2 Adaptation based on parameterization of the selected modalities

Taking a step further from modality selection research has been also conducted into the appropriate parameterization that should happen to a selected modality so as to best suit the needs of each specific user. This is considered a complementary step to the one of modality selection following the pattern “First introduce logic to select the most appropriate modality and then perform another step where the modality itself is adapted to the requirements of the user”.

2.6.3 Fusing modalities

But what happens if more than one modalities are active concurrently? This is not covered both from the perspective of modality selection and from the one of modality adaptation. An architectural approach, which is appropriate for the integration of

modalities like speech and gesture, involves fusing the semantic meanings of input signals. The two input signals do not need to occur simultaneously, and they can be recognized independently. This semantic fusion architectural approach requires less training data, and entails a simpler software development process [114].

As an illustration of the semantic fusion approach is provided by the QuickSet multimodal architecture and information processing flow. QuickSet is an agent-based, collaborative multimodal system that runs on personal computers ranging from handheld to wall-sized [115]. The basic system has been developed in conjunction with a variety of map-based applications, including medical informatics, military simulation and training, 3D virtual-terrain visualization, and disaster management [115], [116]. QuickSet enables a user to create and position entities on a map or virtual terrain with speech, pen-based gestures, and/or direct manipulation. These entities then are used to populate a simulation or other map-based application. The user can create map-based objects by speaking their names and distinguishing characteristics, while simultaneously using a pen to designate information like location, number, and shape.

2.6.4 Guidelines to design for modality selection, adaptation and fusion

An accomplished practice in HCI is the usage of guidelines to support the design of interactive systems. The term “guideline”, in the present context, entails all forms of abstract or concrete recommendations that can be used to design interactive software. Guidelines can be expressed as general and domain independent recommendations [117] or platform-specific style guides [118], [119], [120], [121] or experience-based usability heuristics [122], [123].Guidelines can be embedded in reference manuals and standards, or can be part of the corporate culture and practice of an organization (i.e., customized corporate design wisdom).

The following definition of guideline is adopted in the context of this thesis: “any design and/or evaluation principle to be observed in order to get and/or guarantee the usability of a user interface (UI) for a given interactive task to be carried out by a given user population in a given context” [124]. This section presents a set of

guidelines that were used during the design of the multimodal interaction capabilities supported by this research work [125].

2.6.4.1 Guideline 1: Design for broadest range of users and contexts of use.

Designers should become familiar with users' psychological characteristics (for example, cognitive abilities, motivation), level of experience, domain and task characteristics, cultural background, as well as their physical attributes (for example, age, vision, hearing). An application will be valued and accepted if it can be used by a wide population and in more than one manner. Thus, multimodal designs can aid in extending the range of potential users and uses, such as when redundancy of speech and keypad input enables an application to be used in dark and/or noisy environments.

2.6.4.2 Guideline 2: Take into consideration and appropriately address privacy and security issues.

Users should be recognized by an interface only according to their explicit preference and not be remembered by default. In situations where users wish to maintain privacy by avoiding speech input or output, multimodal interfaces that use speech should also provide a non-speech mode to prevent others from overhearing private conversations.

2.6.4.3 ***Guideline*** 3: Maximize human cognitive and physical abilities.

Designers need to determine how to support intuitive, streamlined interactions based on users' human information processing abilities (including attention, working memory, and decision making) for example:

- Avoid unnecessarily presenting information in two different modalities in cases where the user must simultaneously attend to both sources to comprehend the material being presented [126], [127]; such redundancy can increase cognitive load at the cost of learning the material [126].

- Maximize the advantages of each modality to reduce user's memory load in certain tasks and situations, as illustrated by these modality combinations [128], [129]:
    - System visual presentation coupled with user manual input for spatial information and parallel processing;
    - System auditory presentation coupled with user speech input for state information, serial processing, attention alerting, or issuing commands.

2.6.4.4 *Guideline 4: Integrate modalities in a manner compatible with user preferences, context, and system functionality.*

Additional modalities should be added to the system only if they improve satisfaction, efficiency, or other aspects of performance for a given user and context.

When using multiple modalities:

- Match output to acceptable user input style (for example, if the user is constrained by a set grammar, do not design a virtual agent to use unconstrained natural language);
- Use multimodal cues to improve collaborative speech (for example, a virtual agent's gaze direction or gesture can guide user turn-taking);
- Ensure system output modalities are well synchronized temporally (for example, map-based display and spoken directions, or virtual display and non-speech audio);
- Ensure the current system interaction state is shared across modalities and that appropriate information is displayed in order to support:
    - Users in choosing alternative interaction modalities;
    - Multi-device and distributed interaction;
    - System capture of a user's interaction history.

2.6.4.5 *Guideline 5: Integrate Adaptivity.*

Multimodal interfaces should adapt to the needs and abilities of different users, as well as different contexts of use. Dynamic adaptivity enables the interface to degrade

gracefully by leveraging complementary and supplementary modalities according to changes in task and context. Individual differences (for example, age, preferences, skill, sensory or motor impairment) can be captured in a user profile and used to determine interface settings such as:

- Allowing gestures to augment or replace speech input in noisy environments, or for users with speech impairments;
- Overcoming bandwidth constraints (for example, local direct manipulation replaces gaze input that must be analyzed remotely);
- Adapting the quantity and method of information presentation to both the user and display device.

2.6.4.6 Guideline 6: Be consistent.

Presentation and prompts should share common features as much as possible and should refer to a common task including using the same terminology across modalities. Additional guidelines include providing consistent:

- System output independent of varying input modalities (for example, the same keyword provides identical results whether user searches by typing or speaking);
- Interactions of combined modalities across applications (for example, consistently enable shortcuts);
- System-initiated or user-initiated state switching (for example, mode changing), by ensuring the user's interaction choices are seamlessly detected and that the system appropriately provides feedback when it initiates a modality change.

2.6.4.7 Guideline 7: Provide feedback.

Users should be aware of their current connectivity and know which modalities are available to them. They should be made aware of alternative interaction options without being overloaded by lengthy instructions that distract from the task. Specific examples include using descriptive icons (for example, microphone and speech bubbles to denote click-to-talk buttons), and notifying users to begin speaking if

speech recognition starts automatically. Also, confirm system interpretations of whole user input after fusion has taken place [130], rather than for each modality in isolation.

2.6.4.8 Guideline 8: Provide the appropriate error Prevention/Handling mechanisms.

User errors can be minimized and error handling improved by providing clearly marked exits from a task, modality, or the entire system, and by easily allowing users to undo a previous action or command. To further prevent users from guessing at functionality and making mistakes, designers should provide concise and effective help in the form of task-relevant and easily accessible assistance.

Some specific examples include [131]:

- Integrate complementary modalities in order to improve overall robustness during multimodal fusion, thereby enabling the strengths of each to overcome weaknesses in others;
- Give users control over modality selection, so they can use a less error-prone modality for given lexical content;
- If an error occurs, permit users to switch to a different modality;
- Incorporate modalities capable of conveying rich semantic information, rather than just pointing or selection;
- Fuse information from multiple heterogeneous sources of information (that is, cast a broad “information net”);
- Develop multimodal processing techniques that target brief or otherwise ambiguous information, and are designed to retain information.

2.7 Existing Technologies

2.7.1 The Robot Operating System (ROS)

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

ROS is an important asset in the robotics filed of research because creating truly robust, general-purpose robot software is a hard task. From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own [132].

As a result, ROS was built from the ground up to encourage collaborative robotics software development. For example, one laboratory might have experts in mapping indoor environments, and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, and yet another group might have discovered a computer vision approach that works well for recognizing small objects in clutter. ROS was designed specifically for groups like these to collaborate and build upon each other's work, as is described throughout this site.

ROS is used by students of all ages, from kids interacting with robots in museum exhibits to graduate students learning about the latest solutions to common robotics problems. ROS supports such a wide variety of robots, including low-cost platforms like the TurtleBot and LEGO Mindstorms, hence ROS is especially well-suited to classroom use and rapid prototyping [133].

ROS is an open-source, meta-operating system for robots. It provides the services one would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as Player[1], YARP[2], Orocos[3], CARMEN[4], Orca[5], MOOS[6], and Microsoft Robotics Studio[7].

[1] http://playerstage.sourceforge.net/
[2] http://eris.liralab.it/yarp/
[3] http://www.orocos.org/
[4] http://carmen.sourceforge.net/
[5] http://orca-robotics.sourceforge.net/
[6] http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php
[7] https://www.microsoft.com/en-us/download/details.aspx?id=29081

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server.

The goal of ROS is not to be a framework with the most features but to support code reuse in robotics research and development. ROS is a distributed framework of processes (aka Nodes) that enables executables to be individually designed and loosely coupled at runtime. These processes can be grouped into Packages and Stacks, which can be easily shared and distributed. ROS also supports a federated system of code Repositories that enable collaboration to be distributed as well. This design, from the filesystem level to the community level, enables independent decisions about development and implementation, but all can be brought together with ROS infrastructure tools. However, there are several other goals of the ROS framework:

- Thin: ROS is designed to be as thin as possible so that code written for ROS can be used with other robot software frameworks. A corollary to this is that ROS is easy to integrate with other robot software frameworks: ROS has already been integrated with OpenRAVE, Orocos, and Player.
- ROS-agnostic libraries: the preferred development model is to write ROS-agnostic libraries with clean functional interfaces.
- Language independence: the ROS framework is easy to implement in any modern programming language. It is already implemented in Python, C++, and Lisp, and there are experimental libraries in Java and Lua.
- Easy testing: ROS has a built-in unit/integration test framework called rostest that makes it easy to bring up and tear down test fixtures.
- Scaling: ROS is appropriate for large runtime systems and for large development processes.

2.7.2 FORTH's Ambient Intelligence Network Environment (FAmINE)

An AmI infrastructure aims to support users in carrying out their everyday life activities by offering them an easy and natural way for interacting with the digital services that are provided by the hidden interconnected computing systems. In this respect, AmI environments provide the means to sense and construe the actions that serve the needs of their users in order to offer a personalized, context-sensitive and efficient interaction platform. In an environment where interactions are realized by the confluence of different interconnected computing systems, the organization of the overall system architecture to a well-defined set of distributed software entities is crucial.

FAMINE (FORTH's AMI Network Environment) [134], is a service middleware which has been implemented in the context of the FORTH-ICS AmI Programme, and provides the necessary functionality for the intercommunication and interoperability of heterogeneous services hosted in an AmI Environment. It encapsulates mechanisms for service discovery, event driven communication, remote procedure calls, etc., supporting a plethora of programming languages and frameworks, i.e., the .NET languages family, Java, Active Script, ANCI C++, Python, etc.

2.7.3 Programming Environments and Tools

Choosing a programming language depends on one's language experience and the scope of the application for which the language is needed. While small applications are often created using only one language, it is not uncommon to develop large applications using multiple languages. For example, when extending an application with existing XML Web services, a scripting language may be used with little or no programming effort. For client-server applications, a single language for the entire application based on one's experience is a safer choice. For new enterprise applications, where a large team of developers create components and services for deployment across multiple remote sites, the best choice might be to use several languages depending on developer skills and long-term maintenance expectations.

The .NET Platform programming languages — including Visual Basic .NET, Visual C#, Managed Extensions for C++, and many other programming languages from various vendors — use .NET Framework services and features through a common set of unified classes. The .NET unified classes provide a consistent method of accessing the platform's functionality. All tasks follow the same uniform architecture eliminating the need to learn and master different API architectures to develop different applications.

In most situations, any of the Microsoft programming languages would suffice. Nevertheless, each programming language has its relative strengths and the final language selection should be based on the understanding of the features unique to each language.

2.7.3.1 C#, XAML and WPF

Visual C# (pronounced C sharp) is designed to be a fast and easy way to create .NET applications, including Web services and ASP.NET Web applications. Applications written in Visual C# are built on the services of the common language runtime and take full advantage of the .NET Framework.

C# is a simple, elegant, type-safe, object-oriented language recently developed by Microsoft for building a wide range of applications. It is a language relatively easy to adapt to, given any experience with C and similar languages. It is designed to bring rapid development to C++ programmers without sacrificing the power and control that are a hallmark of C and C++. Because of this heritage, C# has a high degree of fidelity with C and C++, and developers familiar with these languages can quickly become productive in C#. C# provides intrinsic code trust mechanisms for a high level of security, garbage collection, and type safety. Furthermore it supports single inheritance and creates Microsoft intermediate language (MSIL) as input to native code compilers.

C# is fully integrated with the .NET Framework and the common language runtime, which together provide language interoperability, garbage collection, enhanced security, and improved versioning support. It simplifies and modernizes some of the more complex aspects of C and C++, notably namespaces, classes, enumerations, overloading, and structured exception handling. It also eliminates C and C++ features

such as macros, multiple inheritance, and virtual base classes. For current C++ developers, C# provides a powerful, high-productivity language alternative.

Visual C# provides prototypes of some common project types, including:

- Windows Application.
- Class Library.
- Windows Control Library.
- ASP.NET Web Application.
- ASP.NET Web Service.
- Web Control Library.
- Console Application.
- Windows Service.

Windows Presentation Foundation (WPF) is a unified presentation subsystem for Windows, exposed through WinFX, the managed-code programming model for Windows Vista that extends the Microsoft .NET Framework. WPF provides developers with a unified programming model for building rich Windows smart client user experiences that incorporate UI, media, and documents. It consists of a display engine and a managed-code framework. WPF unifies how Windows creates, displays, and manipulates documents, media, and user interface (UI), enabling developers and designers to create visually pleasing, differentiated user experiences. WPF is based on managed code but uses a markup language, Extensible Application Markup Language (XAML), to make building applications much easier for designers.

The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so that other elements of the .NET Framework class library can be easily incorporated.

Extensible Application Markup Language, or XAML (pronounced "zammel"), is an XML-based markup language developed by Microsoft. XAML is part of the Microsoft

Windows Presentation Foundation (WPF). WPF is the category of features in the Microsoft .NET Framework that deal with the visual presentation of Windows-based applications and Web browser-based client applications.

2.7.3.2 WPF and XAML selection rationale

WPF is a second generation GUI and most importantly it makes use of the modern video hardware nearly all machines have. The Windows Forms API was invented in the days when graphics cards were simple and it makes very little use of hardware graphics acceleration. WPF, on the other hand, uses the DirectX rendering engine and this does make use of graphics acceleration. It also means that WPF has access to a wide range of visual effects for full 3D presentation, including meshes, materials, lights, textures and so on.

All of the elements of any UI created using WPF are rendered using vector graphics, rather than bitmaps, which means that they display at the highest resolution the display device supports. It is also a "retained" mode graphics system which makes anything drawn on a form persistent without having to redraw it. For example, if a line is drawn on a form and the form is minimized, the line is still there when the form is restored and the developer doesn't have to redraw it. Retained mode allows the graphics system to optimize the display process.

WPF also implements a better and more sophisticated event system. It also supports UI features that make it easier to implement advanced architectures like MVC. In particular WPF supports data binding which will automatically display and update data bound to UI controls.

WPF is designed to allow you to create dynamic, data driven presentation systems. Every part of the system is designed to create objects through property sets that drive behavior. Data binding is a fundamental part of the system, and is integrated at every layer. Traditional applications create a display and then bind to some data. In WPF, everything about the control, every aspect of the display, is generated by some type of data binding. The text found inside a button is displayed by creating a composed control inside of the button and binding its display to the button's content property.

The most obvious feature of WPF is XAML and it is worth pointing out that even though XAML is central to the way most people use WPF, WPF can be used without XAML. There are .NET classes for each and every WPF component that is available. For example besides the *<Button>* tag there is a Button class within the framework (not to be confused with the Windows Forms Button class). In fact there has to be a framework class for everything in XAML because XAML is just an object creator and initialization language. XAML is always converted to framework objects, properties and methods. This means that the exact same objects can be created and work with their properties and methods without any XAML.

2.8 Discussion

This chapter has provided an overview of various research studies that have been carried out in the context of psychological aspects on the functioning of elderly people. Furthermore it has examined the different multimodal interaction techniques in ambient assisted living environments targeting the elderly and discussed the challenges that the elderly target user group imposes on the design. Moreover, it presented the aspects of human robot interaction in ambient intelligence environments and discussed the different adaptation options that that the multimodal interaction paradigm has to offer. Finally, it discussed existing technologies regarding the human robot and intelligent environment interaction as well as state of the art programming environments and tools in the same context.

Although robotic platforms have been around for quite some time, researchers have been trying to overcome essential and elementary problems that are related to the nature of robotics and their usage in domestic environments. Successful integration of robotic platforms in domestic environments has been the result of a multidisciplinary effort from various scientific fields ranging from computer vision to machine learning and information systems. The challenges of these approaches have been overwhelming while more focus was given to achieving a safe robot behavior in a domestic environment than focusing on the actual human-robot interaction between the platform and its users. However, since the field of robotics has matured

over the last years, a focus shift from the hardware itself to the HRI part of the robot's existence in such domestic environments is becoming more and more necessary.

To this end, the current study focuses on creating a universal solution targeted to be used as the corner stone for building multimodal, elderly friendly, interactive applications that target household robots for elderly users. This thesis provides developers with the necessary technologies, tools and building blocks for creating easy to use elderly-friendly multimodal applications in AAL environments, with particular focus on robotic platforms, thus increasing their level of adaptation to users' needs, and, as a consequence, users' acceptance of these technologies. Using the proposed framework makes these applications inherently friendly to the elder users, while adapting to their needs, the surrounding environment and the context of use. This results in a smooth learning curve, providing increased levels of user satisfaction. Moreover, the use of the proposed framework introduces new interaction modalities such as voice and gestures, enriching and simplifying the interaction experience of the applications.

3 Human Robot Interaction Design

3.1 User groups and requirements

This section aims at specifying the target user groups that were addressed by this research work. Since this research work aims to enable and facilitate the creation of elderly-friendly, multimodal, interactive applications for assistive robots, the target user groups include the application developers who are going to use the proposed development framework to build elderly-friendly multimodal interactive applications, as well as the elderly who are going to use these developed applications. This user base sets specific requirements for the development framework as well as for the applications to be developed, especially in relation to usability, learnability, etc. Each of these requirements encompasses a specific user goal that should be achieved through the usage of the proposed framework.

3.1.1 User Groups

This section identifies the stakeholders addressed by the present research work, analyzes the basic fields of interest and makes an analysis of each user group involved.

3.1.1.1 UG1 – Application Developers

Application developers are involved in the creation and production of elderly-friendly applications for assistive robots. Their work involves both designing (including artwork, graphics and animations) and programming the actual application. In details, there are many stages including creating and designing an application's look (user interfaces) and logic (both interaction logic and functionality), animating UI elements, creating audio in terms of sound effects and speech output, programming, globalization and localization, testing and producing and finally deploying the produced application. Typical activities of application developers may include:

- Development of designs and/or initial concept designs for applications including application functionality
- Generation of application logic, scripts and UI animation storyboards
- Creation of the visual aspects of the application at the concept stage

- Usage of 2D or 3D modelling and animation software, such as Maya, for the creation of 2D/3G application models and graphics at the production stage
- Production of the audio features of the application, such as music, speech output and sound effects
- Development and integration of different interaction modalities including touch, voice in terms of recognition and synthesis, gestures and other modalities such as facial expressions (used as output), hardware switches and buttons, head and gaze trackers, actuators etc.
- Development of reasoning components capable of inferring the necessary decisions to support the application's logic and functionality
- Development of adaptation components capable of adapting the applications and the whole interaction experience both to the preferences and needs of the end users as well as to the context of interaction (both semantic and environmental).
- Programming the actual application using programming languages such as C#
- Quality regression testing of applications in a systematic and thorough way to find problems or bugs and recording precisely where the problem was discovered
- Provision of solutions to complex technical problems that occur within the application's production
- Dissemination of knowledge to colleagues, clients, publishers and end-users
- Understanding complex written information, ideas and instructions
- Working closely with team members to meet the needs of projects
- Planning resources and managing both the team and the development process
- Performing effectively under pressure and meeting deadlines to ensure the applications are completed on time

In this thesis, the user group "Application Developers" (UG-1) consists of those developers who are responsible for the development of the applications' UIs, interaction logic and functionality, adaptation and reasoning components, as well as the applications' logic scripts. Furthermore, this group includes developers

responsible for programming the actual applications using programming languages while working closely with team members to meet the needs of projects.

3.1.1.2 UG2 – The Elderly

The second target user group is the elderly user group. The elderly population is a very heterogeneous group, displaying a wide variety of characteristics and abilities. For example, many older people are physically fit and may share similar user characteristics as younger adults. On the other hand and particularly as people reach old age, some elders are affected by vision, hearing and memory loss, in addition to dexterity problems caused by arthritis, for example. This ageing process affects an individual's ability to interact successfully with a standard graphical user interface. The process of interacting with touch user interfaces is even more difficult, as diminished visual acuity makes it very difficult to see items on relatively small interfaces and cognitive impairment increases the difficulty of conceptualizing and navigating an interface. This makes it difficult to design technological applications that suit this entire diverse user group. Multimodal interfaces, however, offer a flexible design, allowing people to choose between combinations of modalities, or to switch to a better-suited modality, depending on the specifics of their abilities, the task and the usage context. This flexibility is a key feature which makes multimodal interfaces ideal for mobile computing for the elderly [135].

3.1.2 User Requirements

System Requirements capture the intended functionality and behavior of the system so as to drive architectural decisions and validate the architecture. In the context of this research work requirement, requirements were partitioned into *functional requirements*, *non-functional requirements* and *interaction requirements*.

Functional requirements are associated with specific functions, tasks or behaviors that the system must support. In other words, functional requirements clarify the functionality required by the system towards supporting effectively (an agreed set of) user tasks

Non-functional requirements are constraints on various attributes of these functions or tasks associated with the user's goals. Non-functional requirements specify the required behavior of the system towards supporting efficiently the user goals. It can be helpful to think of non-functional requirements as adverbially related to tasks or functional requirements: how fast, how efficiently, how safely, etc., is a particular task carried out through a particular system.

Interaction requirements are used to identify the quality of the human robot interaction so as to achieve the goals set by this research work.

3.1.2.1 Requirements collection methodology

The methodology used for collecting and elicitating requirements includes browsing exhaustively the currently published literature (paper or electronic), in reports, journals, conference proceedings, and books. Keywords and key subject titles, relative to the problem area and a set of specific research questions that would drive our analysis efforts was formulated and used to initiate and drive a search that was carried out. The **first phase** concerned an initial literature review, which was followed by reviewing relevant deliverables from two currently ongoing research project funded by the European Commission, namely HOBBIT and RAMCIP. In both projects an extensive user requirements collection methodology was followed and the outcomes were published in well established conferences and journals.

Furthermore, the developers' user requirements were elicitated by means of interviews and close observation of a group of developers in real life action, in the process of producing elderly-friendly applications for assistive robots in the context of the two aforementioned research projects.

3.1.2.2 Functional Requirements

Functional requirements capture the functionality required by the system towards effectively supporting (an agreed upon set of) user tasks. In other words, functional requirements define what a system is supposed to do, i.e., "**system shall <do requirement>**". For the purposes of this study, the word "**system**" represents the

entire proposed framework to support the creation of elderly friendly-multimodal applications for assistive robots.

3.1.2.2.1 UG1 – Application Developers

Under the perspective of elderly-friendly multimodal interactive application development for assistive robots, the following set of requirements was identified as necessary for application developers:

- The system should maximize the potential for incorporating heterogeneous technologies
- The system should support different interaction modalities in an efficient way and support easy integration of software components
- The system should enable modality integration both in the scope of application dialogues as well as in the scope of higher application logic
- The system should provide sensible programming abstractions
- The system should enable the implementation of higher-level functionalities by composing basic building blocks and / or services
- The system should provide a flexible model for representing and using the context of the robotic platforms and the AmI environment towards supporting the elderly in their everyday lives
- The system should provide libraries and tools for developing and deploying services
- The system should provide the necessary functionality for integrating the functionality offered by both the robotic platform and the surrounding AmI environment
- The system should provide basic building blocks, UI elements, components and dialogues for building multimodal interactive elderly-friendly applications
- The system should provide the necessary reasoning and adaptation components to enable the tailoring of the developed applications to the needs and preferences of the users as well as to the semantic and environmental context of interaction

- The system should provide libraries and tools for developing and deploying applications suitable for elderly users
- The system should provide the means for simulating the functionality of the robotic platform and the AmI environment supporting elderly users into their everyday lives
- The system should provide libraries and tools for developing and deploying multimodal interactive applications to help the elderly in their everyday lives.

3.1.2.2.2 UG2 – The Elderly

Using the aforementioned methodology, the functional requirements for the elderly user group were collected as presented in the following table. Each functional requirement presented in the table is numbered, has a description, the type of functionality that is targeted and a level of importance. The scaling used for representing importance is (low, medium, high). This scale was used in order to prioritize each requirement in the context of development and assist the evaluation of the final product.

No.	Type	Requirement	Importance Value
1	Robotic assistant communication	The system should be operated with voice commands/ gestures.	High
2		The system should react to name.	High
3		The system could use honorific forms and tell jokes.	Medium
4		Robotic assistant could be operated with touch screen.	Medium
5		Robotic assistant should talk back to the user.	Medium
6		Robotic assistant is able to take part into dialogue interactions with the user in order to complete required tasks.	Medium

7		Robotic assistant could be controlled directly from the touchscreen it carries without the need to engage in a dialogue with the user.	Medium
8		The robotic assistant should continuously listen to the user for commands and respond to simple commands user give.	Medium
9		The robotic assistant should talk back to the user regarding its current task / state and reply to simple questions (e.g. what time is it?)	Medium
10		The robotic assistant should comprehend and respond to simple gestures that the user makes.	Medium
11		The robotic assistant should provide positively affective impact to the user regarding his emotional state (actions).	Medium
12		Robotic assistant should be operated with remote control.	Low
13	Robotic assistant appearance	Robotic assistant should have positive emotional facial expressions.	Medium

Table 1: Functional Requirements

3.1.2.3 Non-functional requirements

Although the provision of the above mentioned functionality is of high value to both the developers and the end-users, they might end up disliking the system if some of their goals (e.g., to perform tasks as quickly as possible; to stay away from big mistakes, to be consistent) are violated while using the system. The identified non-functional requirements are documented as follow:

<keyword>: <non-functional requirement>

Non-functional keywords include, but are not limited to: Usability, in terms of Learnability, Efficiency, Memorability, Errors, Satisfaction; Security; Reliability; Maintainability; Portability; Extensibility; Reusability; Resource utilisation; Operability; and Accessibility.

- **<Learnability>:** It is essential for each application incorporated to the robotic assistant to be developed to require a minimum level of learning from the user side. To this end the provision of easy to use, self-descriptive and intuitive User Interfaces is a fundamental requirement. This is extremely important in the case of this research work so as to cope with the diversity of the target users' population (elderly user group).
- **<Efficiency>:** Efficiency is measured as the resources expended by the user in relation to the accuracy and completeness of goals achieved. This is a very important issue when designing to assist users performing daily activities. Technology acceptance does not improve when providing impressive of fancy systems, but when offering more efficient ways of performing specific activities.
- **<Memorability>:** This requirement implies the provision of applications containing simple and self-descriptive operations not relying on the user's ability to remember sequences of actions. This may be achieved by allowing the user to perform operations in a way similar to actions performed in the context of his every day activities. This is important for all the areas targeted by this research work, but especially for modality selection and modality adaptation.
- **<Errors>:** The ability to minimize errors and recover from system and user errors is vital. The applications should avoid the typical attitude of living the impression that the user is responsible for application errors and at the same time be in position to recover "silently" without the need for user input. In the same context, user errors should be foreseen and avoided through the appropriate run time user assistance.

- **<Satisfaction>:** Offering satisfaction to the user accessing an interactive application is considered as a primary requirement. More specifically, this is achieved by offering support for recovering from user errors and furthermore via offering an overall feeling of smooth operation and facilitating the user to perform complex operations through an intuitive and usable interface.
- **<Security>:** Security is considered a major requirement especially for applications that are interconnected with smart environments where personal user data are at stake. In such environments where cameras, intensive user profiling, sensitive user data (bank accounts, credit cards) etc. are present there is always an issue of compromising sensitive data. The usage of such data in a way that sensitive user information are shield for unauthorized access is most important for allowing these technologies to spread and get accepted by people in their everyday life. In this context, sensitive user information should be stored by the system only in cases where the computing environment is isolated and secured. It is important to keep such information confidential and don't submit them over the network.
- **<Reliability>:** Although all the above requirements are important, nothing can be achieved by relying on an unreliable system. Fault tolerance, consistent operation, tolerance to loss of network connectivity, ability to recover for power loss, etc. are major requirements. All these issues should be identified and solved transparently. It is not wise to depend upon the average user on identifying and manually solving such issues especially in environments created for non IT experts such the ones presented in the context of this research work.
- **<Maintainability>:** It is traditionally derived from the field of human computer interaction that users should pay time for performing their tasks rather than managing and maintaining the interface offered for performing these tasks. To this end, it is important to create applications that are self-maintained with minimum feedback from the user.
- **<Extensibility>:** The provision of a framework that offers the option for the incorporation of new applications is essential. This although partially

addressed by this research work requires further investigation from the field of standardization for offering common hardware and software standards.

3.2 The RAMCIP platform

RAMCIP (Robotic Assistant for Mild Cognitive Impairment Patients at home) is a Research and Innovation Action (RIA) funded by the EC in the context of the HORIZON2020 Programme, with the aim to research and develop a novel domestic robotic assistant for MCI and early AD patients. RAMCIP is going to research and develop a novel robot that can provide proactive and discreet assistance to elderly people with MCI in their own home, to support their independent living and quality of life.

RAMCIP aims to research and develop real robotic solutions for assistive robotics for the elderly and those suffering from Mild Cognitive Impairments and dementia. This is a key step to developing a wide range of assistive technologies. The objective of the project is to adopt existing technologies from the robotics community, fuse those with user-centered design activities and practical validation, with aim to create a step-change in robotics for assisted living.

The RAMCIP vision corresponds to future service robots for assisted living environments that can provide safe, proactive and discreet assistance in significant aspects of the user's daily life, ranging from food preparation, eating and dressing activities, through to managing the home and keeping it secure. At the same time, the robot should help the user maintain a positive outlook and also to exercise his/her cognitive and physical skills. RAMCIP will also work towards future robots that help the users to perform exercises as part of their assistive work, thus embedding exercise in their daily routine.

RAMCIP follows an inter-disciplinary approach in studying a series of aspects ruling Human-Robot and Robot-Environment Interaction in ambient assisted living environments, toward advancing the way that service robots coexist with humans, not only as assistive computer-based servants capable of obeying commands, but as true

companions, living and evolving along with the human, towards advancing Human-Robot Interaction of the present into Human-Robot Coexistence of the future.

The proposed system's core will be an autonomously moving service robot with an on-board touch screen and a dexterous robotic hand and arm. Following the paradigm of relevant research projects, the robot of RAMCIP will first of all build upon existing knowledge so as to include advanced versions of features already researched and developed in the past, such as entertainment functions with emphasis on enhancing the physical and cognitive state of older and early MCI persons, providing also access to social media and telepresence-based communication with relatives, friends or caregivers, as well as basic assistive functionalities such as bringing objects to the user and picking up objects from the floor. Having features like the above as a starting point and a thorough investigation of user needs as a basis, the project will aim to advance the state of art in domestic service robotics.

Among the primary objectives of the RAMCIP system is to develop novel methods for personalized and adaptive multimodal Human Robot Interaction, allowing the robot to automatically switch between different interaction modalities or fuse them, ensuring the provision of optimal HRI on the basis of user behavior and context. This HRI will be strongly augmented through novel empathic communication channels and novel human affect recognition methods based on analysis of the user pose and gestures. Furthermore, the robot's behavior will be subject to the user's affect, trying to induce positive emotions when needed. The envisaged service robot will include novel interfaces based on augmented reality concepts, projecting information and highlights over locations in the room and objects.

To this end, the majority of the functionality offered by the FIRMA framework is going to be used for the Human Robot Interaction parts in the context of the RAMCIP project. A multimedia suite of elderly-friendly multimodal interactive applications will be based on the FIRMA framework to deliver the robot's functionality to the end users. The contribution of the FIRMA framework is crucial because it provides all the necessary functionality to adapt to the needs and preferences of the elderly user group. The dynamic adaptation capabilities of the framework in terms of adaptive component hierarchies will facilitate the tailoring of the HRI to the continuously

changing needs of MCI and early AD patients, while respecting their preferences and adapting to the changing context of use. Furthermore, the adoption of the adaptive cascading style hierarchies will maximize the adaptation capabilities of the system while delivering a uniform look and feel throughout the graphical interfaces that the robot will offer. Moreover, the reuse of the ready-made framework elements, components and dialogues, will contribute to a smoother learning curve. This is a very important aspect regarding robotic platforms and the elderly user group, especially for the MCI target user group of the RAMCIP project, since the users will be able to reapply the acquired knowledge throughout their interaction with the developed applications and the platform. Using a uniform look and feel as well as reusable components across all developed applications contributes to increased ease of use and provides the ground for higher acceptance levels of these technologies. Additionally, the inherent support for multimodality will contribute to a more natural interaction between the robotic platform and its users. This also contributes to the ease of use, since users will be able to interact with their voice and gestures as additional modalities to the robot's onboard touch screen. Finally, the integration capabilities of the FIRMA framework with respect to the ROS operating system will provide to the developed applications access to the functionality that will be provided by the robot while enabling the use of the various robot sensors to build more complex and smarter robot behaviors. The inherent support for the ROS operating system will provide access to the augmented sensing capabilities of the robot that will maximize the potential of the robot's behavioral models.

3.3 Scenarios

3.3.1 A day with Olive

Olive is an 80 year old proud grandmother of Ben (20yo) and Ellie (10yo). She lives alone but her grandchildren who live a couple kilometers away from her house visit her twice a week. She lives in a spacious apartment which has a kitchen, a bathroom, and a comfortably big area that contains her bed, sofa and closet as well.

She is unfortunate to suffer from mild hip pain due to a past fall which has resulted into mobility impairment issues and most of the time she has to use a walking

aid to be able to walk with confidence. She has a very mild cognitive decline that is consistent with her old age. Her memory doesn't serve her very well and as a result she might forget some things from time to time. She wears glasses because her eyesight is limited due to farsightedness, but she can move freely and safely in the house. She has several age related blood disorders for which she takes the appropriate medication.

3.3.1.1 Morning

Olivia decides to leave her bed. She sits on the edge of her bed and tries to stand up. She is in pain and afraid that her legs won't be able to support her and asks Eva, her mobile domestic robotic platform, to bring her walking aid (asking for her walking aid means that she is in pain, hence this is a bad day regarding her hip and hence Eva should remind her to take a painkiller with her breakfast).

It is 7am and Olivia has just woken up. Eva greets her, notifies her of a weather change which resulted in an extended temperature drop and brings her robe. Eva has logged that every time the temperature drops below 15 degrees, Olivia feels cold and sets the temperature to the more comfortable level of 20 degrees when she wakes up to heat up the house. Eva has proactively and automatically set the temperature to 20 degrees for Olivia (based on her habits) and kindly informs her of that action and asks her to confirm that this was what she wanted. Olivia responds negatively and Eva prompts her to use the platform mounted touchscreen (which changes to reflect a thermostat control) to set the room temperature to a different level.

Olivia says good morning, Eva responds and asks if she needs help. Olivia kindly asks her to turn on the TV (she wants to watch the morning news on her favorite TV channel). Eva knows that Olivia had been watching a different channel last night and asks her if she wants to turn on the TV to the channel she was watching yesterday or to the one she usually watches in the morning. Olivia responds that she want what she usually watches in the morning and Eva acknowledges that and asks her if she needs anything more. Olivia verbally responds negatively. Olivia asks Eva about the appointments of the day and Eva reminds her that later this morning she has a scheduled appointment to go shopping with her friend Phoebe.

Olivia always takes her morning medication before breakfast and Eva makes sure she doesn't forget. Eva gives the morning medication box to Olivia and prompts her to acknowledge the pill intake by touching the corresponding icons on the platform mounted display which changes to reflect the medication intake task. Eva detects that Olivia's reading glasses have been left on her nightstand (RFID) and informs her that she will approach her further while making the images on the screen bigger because she isn't wearing her glasses and can't see clearly where to touch on the screen. The images on the screen get bigger to help Olivia. Eva asks if Olivia wants her to bring her reading glasses.

After the morning medication, breakfast is in order. Olivia has several blood related medical conditions and her diet has been adjusted accordingly by her physicians. Eva reminds Olivia what is on her dietary menu for breakfast while showing it on the kitchen display and helps her decide on alternative nutritionally equivalent foods based on her most (or least) frequent choices (or favorites).

Eva knows that Olivia has a scheduled appointment, approaches her and suggests her to do some shopping on her way home. Olivia agrees to do the shopping. Eva knows Olivia's dietary plan of the week so she has pre-included all the necessary items in the shopping list. Eva presents a screen to Olivia containing a list of potential things to buy as well as the current shopping list. Olivia can move the things she wants between the lists.

The doorbell rings and Eva approaches Olivia while showing on the tablet display the video feed from the front door. Olivia tells Eva that she isn't wearing her glasses and Eva approaches even more while maximizing the video window taking advantage of all screen real estate. Olivia recognizes her friend Phoebe at the front door and asks Eva to let her in the apartment. Eva brings Olivia her purse and asks her to acknowledge that she is taking her wallet, her mobile phone, her jacket and the keys of the apartment with her while informing her that the bathroom window has been left open and prompts her to close it. Eva asks whether she wants to send the list to Olivia's mobile phone or she prefers to print the list for her. Although Olivia has a smartphone and she can use it to see the shopping list on the screen, she

prefers the natural touch of paper and Eva knows that, hence she printa the shopping list on paper for Olivia.

While Olivia is away shopping, two people ring the front door bell and Eva records a video from the front door camera. One of them is Olivia's best friend Charice and Eva informs her that Olivia is not at home. In addition, Eva logs all unanswered landline calls and voice messages. When Olivia returns home, Eva will prompt her to review the recorded videos and in the case of Charice, Eva prompts Olivia to call her back by showing the contact photo on the platform display (click to call). What is more, Eva shows a detailed call log for the time period Olivia was outside and a list of her voice messages (playback on touch or voice command). Furthermore, while Olivia is away, Eva turns off all electric appliances, lowers the room temperature to 17° and takes the mail that the postman put in the mailbox and places them on the table beside the couch which is where Olivia expects to find it when she returns home. Furthermore Eva tracks Olivia through her cellphone in order to detect when she is close to home to restore the temperature to a more comfortable level and turn on the heater for Olivia's afternoon shower. Furthermore Eva patrols the house while Olivia is away to monitor for potential unwanted intruders as well as to pick up objects that have fallen on the floor and pose a potential hazard as Olivia might trip over them and fall down.

Olivia returns home with the shopping. Eva approaches her and helps her move the shopping to the kitchen. Eva presents to Olivia a screen containing the shopping list items and asks her to select the things that she has bought. If Olivia bought extra stuff, she uses the add button to add the extra items. Eva knows that Olivia is tired and asks her two questions "Did you buy all the things on the list?" and "Did you buy any extra things I need to know about?" in order to save her the trouble of updating the shopping inventory manually. The different screens can adapt according to whether Olivia is wearing her glasses or not. The text to speech volume adapts to the environment noise and the distance between Eva and Olivia.

3.3.1.2 Noon

Olivia returns home having gone to the supermarket and Eva approaches her, takes the groceries (Olivia puts it on the back mounted cart) and follows her to the kitchen. There she guides Olivia through the update of the house's food inventory. Eva calculates that there are too many food categories to show on one screen and changes to an alternative layout. Moreover, she detects that Olivia makes several mistakes during the food input (she is standing and her hands shake) and asks her to change to the swabbing mode. In addition, Eva adjusts the brightness of the screen to compensate for the direct sunlight that comes through the kitchen window. Olivia touches the photos of the products on Eva's display to update the available quantities so that Eva will be able to assist her when deciding what to cook according to the availability of the ingredients. Eva automatically strikes off all the products that Olivia bought from the shopping list. Products with NFC tags are automatically recognized and Eva informs Olivia about which products have been updated and asks her to update manually the rest.

Olivia asks Eva for a recipe according to her dietary plan for the day and the availability of products in the house. Eva proposes two recipes, one of low and one of medium complexity. Olivia feels tired from shopping and chooses the easy one. Eva shows all the required ingredients on the kitchen's screen and starts reading the cooking instructions one by one according to Olivia's feedback. When Eva detects that the recipe is almost finished, she starts laying the table to save Olivia the trouble and time. When the oven timer exceeds the recommended cooking time, Eva informs Olivia that the food might be ready and prompts her to check it and switch off the appliance. Eva prompts to take a picture of Olivia to send it to her grandchildren.

Olivia starts to eat when she remembers that she should take the noon medication with her food. She kindly asks Eva for the appropriate medication box and a bottle of water. Eva would have reminded Olivia about the medication, had she forgotten about it. Eva prompts Olivia to acknowledge the medication intake on the platform's screen. Olivia can't reach the screen due to her fatigue and her relaxed body posture and asks Eva to approach her further or she uses a verbal/gestural acknowledgement. Eva reminds Olivia that she has left the gas on and informs her that

she will switch it off for her because she knows that she is tired. Moreover, Eva asks her if she wants to turn on the kitchen TV to her favorite show.

3.3.1.3 Afternoon

Olivia finishes eating, leaves the kitchen while telling Eva to switch off the TV and enters the living room. Eva knows that the TV show that Olivia was watching in the kitchen is not over yet and asks her if she wants to turn on the TV to continue watching the show. Olivia does not respond and Eva asks her again after increasing the volume.

Olivia is sitting on the living room couch lost in her thoughts. Eva approaches her after 10 minutes and prompts her to have a video chat with her grandchildren. Eva displays the contact information on the onboard display and prompts Olivia to call them. During the call, Eva prompts Olivia to send the picture that she took earlier to her grandchildren to invite them over for cookies.

3.3.1.4 Night

Olivia complains to Eva that she is not feeling well. Eva immediately approaches her while turning down the TV volume and closing the curtains to reduce sunlight. She tries to calm Olivia down while reassuring her that she is there to help her. She displays the family doctor's telephone on the onboard screen (call with a click for instructions regarding counter-reactions with her prescribed medication). Moreover, Eva brings the thermometer to Olivia (or the robot has a thermometer embedded in one of the fingers and hence can check for her temperature or in a respective situation her heart rate or even maybe her blood pressure/glucose level to give these info to the family doctor if Olivia decides to call him). Olivia feels better and tells Eva not to call the doctor but to bring her a painkiller instead. Eva reminds Olivia about the dosage instructions, suggests that Olivia rests and prompts her to cancel her evening meeting with her friends.

It is already past Olivia's bedtime and Eva helps her to go to bed by bringing her walking aid. She informs Olivia about a further temperature drop during the night and prompts her to use another blanket. She also tells her that if she needs her during

the night, she can call her by pressing the button on her nightstand. Eva will be charging during the night as well as patrolling the house for security and housekeeping purposes. Olivia asks Eva to wake her up at 10am next morning and Eva acknowledges.

3.3.1.5 Envisioned applications to assist Olivia

3.3.1.5.1 App1: "Bring me…" application:

- User can verbally ask for objects, e.g., "Bring me my walking aid"
- Takes input from appointments app to automatically bring necessary objects and ask confirmation for others
- Can move things around at predefined times/places

3.3.1.5.2 App2: Medication Supervisor application

- Gets input from the "Bring me" task, if the object is "walking aid" infers that the user is in pain and adds painkillers to the day's medication
- Provides input to the "bring me" application
- Supervises medication intake task

3.3.1.5.3 App3: Weather application

- Monitors the weather and provides input to the bring me application

3.3.1.5.4 App4: Home control application (temperature)

- Monitors user's preferences according to time of day, season, weather, etc.
- Automatically sets the temperature to prevent user discomfort and notifies the user
- Automatically sets the temperature to a predefined lower setting when the user is away
- Automatically sets the temperature back to user setting when the user is close to the apartment (GPS tracking)

3.3.1.5.5 App5: Home control application (TV)

- Monitors user's preferences according to time of day
- Notifies user if last watched channel differs from the usually watched
- Turns off appliances when user is away

3.3.1.5.6 App6: Appointments application

- Manages user's appointments

3.3.1.5.7 App7: General adaptation

- When user isn't wearing glasses, robot approaches
- When user isn't wearing glasses, robot makes images bigger
- When user isn't wearing glasses, robot uses the "Bring me" app to prompt the user to activate the "Bring me – my glasses" scenario.

3.3.1.5.8 App8: Cooking support application

- Helps user choose breakfast according to diet
- Can print shopping list or send it to mobile phone according to preferences
- Adds to shopping list necessary items and informs the user

3.3.1.5.9 App9: Front door intercom application

- Robot adjust position from user to display video feed.
- Video screen can be automatically maximized
- Can open door

3.3.1.5.10 App10: Home control application (Doors)

- Can be used to open doors

3.3.2 Nick the software engineer

Nick is a senior software engineer in a small company with experience in the development of robotic applications for the elderly and people with disabilities. Nick's company is a software integration company. It uses a specific robotic platform for developing its services and the main tasks is writing applications, integrating modalities and interconnecting applications and modalities to the robotic platform. The company uses third party libraries for several functions of the robotic platform such as object recognition and manipulation. Recently, the company has made a decision to integrate the robotic applications it offers with smart environments, while also extending the usage of applications to cover a greater population including people with disabilities. This was a decision that Nick should support at a technical

level. His job is to find the appropriate solutions to a number of issues that should be addressed in order to make feasible the envisioned extensions to the company's activities. Nick has spent some time researching the requirements of such an approach and has concluded that there are several obstacles. More specifically, in terms of modality integration, targeting a wider public may require to integrate additional modalities to the platform such as speech recognition and gesture recognition modalities. Furthermore, there is the issue of providing the appropriate interface to each user that from his perspective would result into having to implement and maintain different versions of the same application. Another issue is the integration with the environment. Nick has learned from his past experience that having services over TCP-IP is a headache that could result into many connections and non-formalized communication between peers. Finally, there is the major problem of logic over who decides what and how the whole interaction is orchestrated. Some form of external rules-based logic is required. Nick knows that this is too much to handle for his development team, so he decides to make some research regarding existing automations and frameworks that could provide solution for at least some of these issues. During his research he is frustrated by the fact that many different solutions exist, but in most of the cases only a very small part of his requirements are covered. Integrating a number of third party libraries and possibly one for each different issue to be solved is a solution, but it is obvious to Nick that it will not result into easily maintainable products. It is important to him that the required solution to his problem be robust, time lasting and capable of covering multiple requirements. "What if we use a technology that will suddenly get discontinued in the future?" Nick thinks. While browsing several resources he finds the FIRMA web site. It is a project aiming to reduce development time by at least 50% while also promising seamless integration with ROS, a number of modalities, adaptation support and environmental awareness. The project claims that in some cases only scripting the logic of an application is enough. Being skeptical, he decides to take a look at some of the training material available, to understand the concept. He notices that FIRMA allows the development of GUI components for robotic platforms that are compatible with existing infrastructure, while also supporting a variety of modalities and environmental sensors. He also realizes that FIRMA has an IDE available to create new UI applications by just scripting

their logic. Furthermore, it is fully extendable as developers can modify or add new controls and screens to best suit their needs. Additionally, he realizes that the logic of the application can also be affected by environmental input, thus achieving environmental awareness. Furthermore, FIRMA has a ready to use middleware and the appropriate middleware bindings with services so as to automate the process of listening to events from the environment. At the end of the day, he is convinced to give it a shot, so he sends an email to his development team to organize a meeting. The electronic engineering team will take over the job of integrating existing and new sensors to the test lab. The hardware administration team is going to integrate the middleware into to the test lab and the robotic platform and make sure that the appropriate sensors to support the FIRMA's offered modalities are installed to the robotic platform. Finally, the application development group is going to create some UIs using FIRMA and will write the applications and adaptation logic. The software development team quickly understands that they can control how the UI adapts in response to the environment and the user without writing extra code. On the second week of development, the team is ready to test the first FIRMA powered scenario. Everything goes quite well in the mini demo the team is running for Nick. Nick thinks that "This is a very good start and what's more important is that using FIRMA we can scale fast and be professional at the same time."

3.4 UI Design

The design of the user interface of FIRMA dialogue controls followed five main stages:

- Low fidelity paper based prototyping of different design alternatives to investigate alternative appropriate UI representations.
- Low fidelity power point based prototyping by means of mockups of some of the UI paper based prototypes in order to assess their functionality.
- Enumeration of different design alternatives to cater to the particular requirements of the users and the specific context of use.
- Encapsulation of the design alternatives into appropriate abstractions and integration into an adaptive component hierarchies.
- Development and documentation of the design rationale that drive the run-time selection between the available alternatives.

The following section presents the design outcomes, including the adaptive component hierarchies, the design space populated by the produced physical designs and for each adaptive component in the hierarchy a design rationale recording its run-time adaptation logic based on user- and context-related parameters.

Figure 1: Low fidelity paper and Power Point based prototyping of the FIRMA framework's UI controls and dialogues

Figure 1 shows the different low fidelity paper and power point prototyping stages that led to the design selection of the FIRMA framework UI elements and dialogues.

3.4.1 Container styles

Two generic template styles were designed (see Figure 2). The linearized template style contains all the containers in a linear form. On the other hand, the columns

template style has three alternative styles where top and bottom navigation are placed on the top and bottom positions and the centered container is split in two, three or four columns respectively.

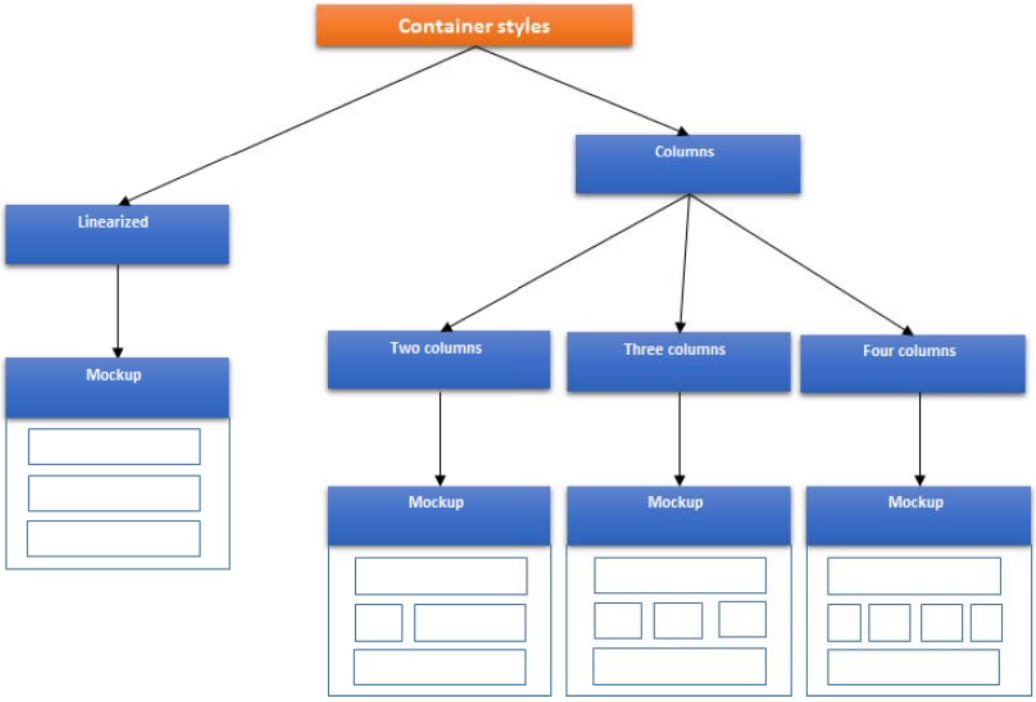


Figure 2: Container alternative styles

Container size constitutes another significant aspect that is associated with the screen size in which the UI application will be presented. As it is presented in Figure 2 and Figure 3, the container size may be resized according the device screen size in order to cover the optimum screen real estate.

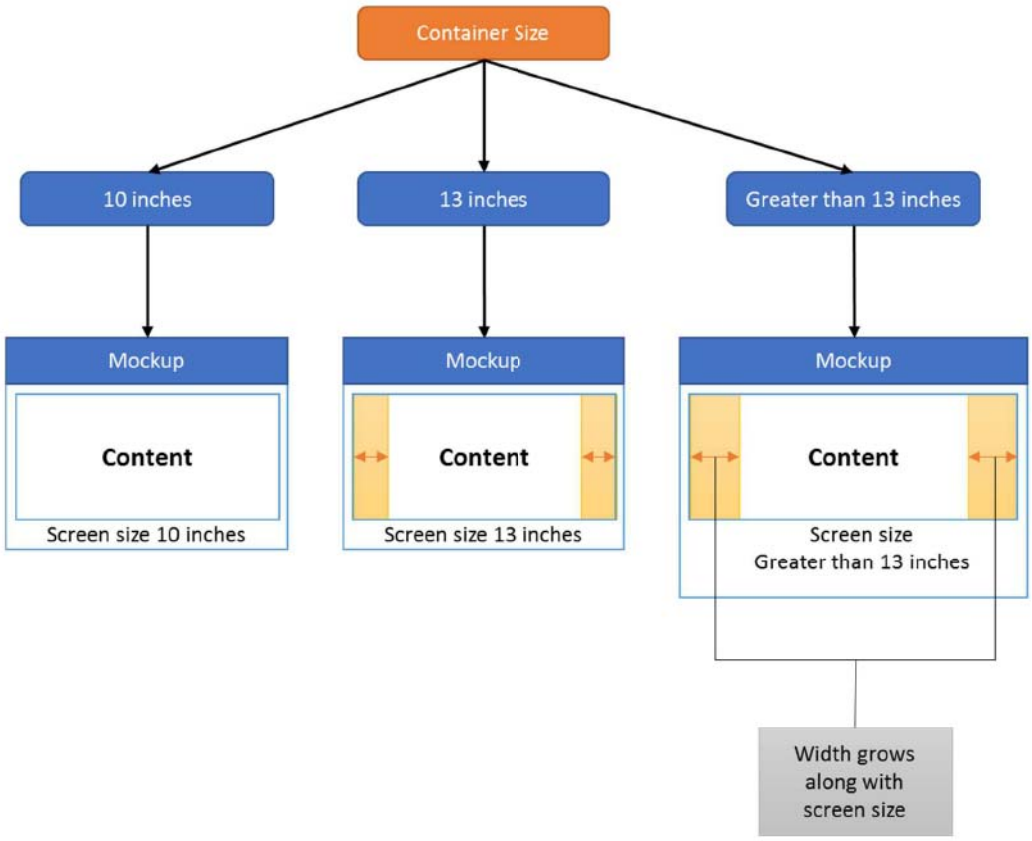


Figure 3: Container alternatives according to device size

When the screen size is up to 10 inches, the content covers all the surface of the screen, whereas for screen sizes from thirteen inches and over, the content is centered while leaving a small unexploited area on the left and right, in order to maximize the readability of the contents. For screen sizes greater than thirteen inches, the width of the empty area on the left and right of the content is adjusted according to the screen size.

3.4.2 Links – Buttons alternatives

Buttons and in some case also links comprise maybe the most used objects in any application. Several alternatives were designed in order to offer effective interaction. As shown in Figure 4, links remains the same and the only difference concerns the background and text color. Generally, links are presented in blue. However, several different color combinations for low vision users and user with color blindness were produced based on (Arditi, 1996).

According to (Arditi, 1996) there are three basic guidelines for making effective color choices that work for nearly everyone:

1. Exaggerating lightness differences between foreground and background colors and avoiding using colors of similar lightness adjacent to one another, even if they differ in saturation or hue (see Figure 4).

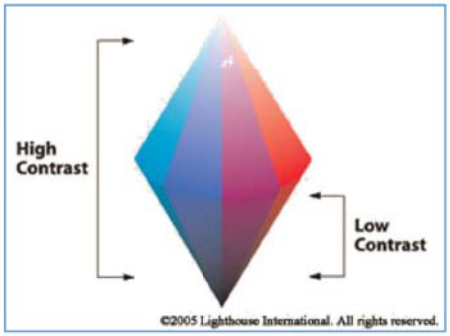


Figure 4: Color contrast

2. Choosing dark colors with hues from the bottom half of this hue circle against light colors from the top half of the circle (see Figure 5). Avoiding contrasting light colors from the bottom half against dark colors from the top half. For most people with partial sight and/or congenital color deficiencies, the lightness values of colors in the bottom half of the hue circle tends to be reduced.
3. Avoid contrasting hues from adjacent parts of the hue circle, especially if the colors do not contrast sharply in lightness.

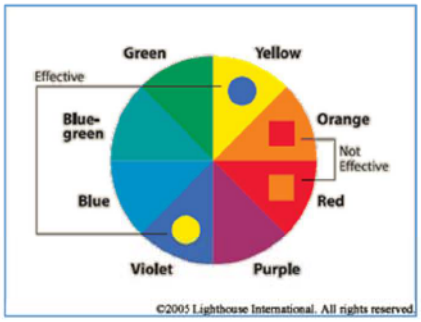


Figure 5: Color wheel

Following the above guidelines, four different color sets were chosen in order to generate correct contrast for all users that face visual impairments: blue link on

white background, blue link on yellow background, orange link on blue background and green link on black background. In order to make a link more visible, the 'on focus' link artifact was designed (see Figure 6), in which a border with the text color is shaped around the link.

A button representation was designed too in order for links to be presented as buttons to be used in a tablet pc. In a tablet pc, the links are designed to be presented as buttons in order to be more "touchable".

Four alternative depictions were designed for buttons, containing different color combinations that produce correct color contrast for users that face visual impairments. The alternative design artifacts include buttons with black color text on light gray background, white color text on blue background, blue color text on yellow background, and black color text on yellow background.

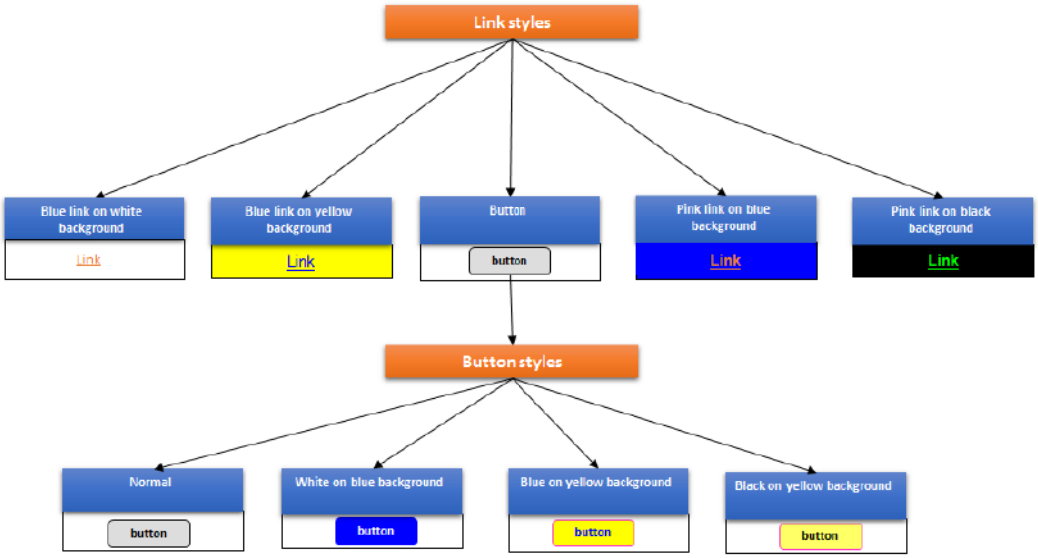


Figure 6: Link – Button representations

3.4.3 Textboxes alternatives

According to previous the section, the combination between text colors and background colors has to produce efficient contrast. This issue is applied to textboxes too, because when a user types text in a textbox, the application has to offer efficient color contrast between the textbox background and the color of the text being typed. According to this issue, three alternative textboxes (on focus) were designed and

rationalized except from the default textbox (see Figure 7). These are yellow text on black textbox background, orange text on blue textbox background, and blue text on yellow textbox background.

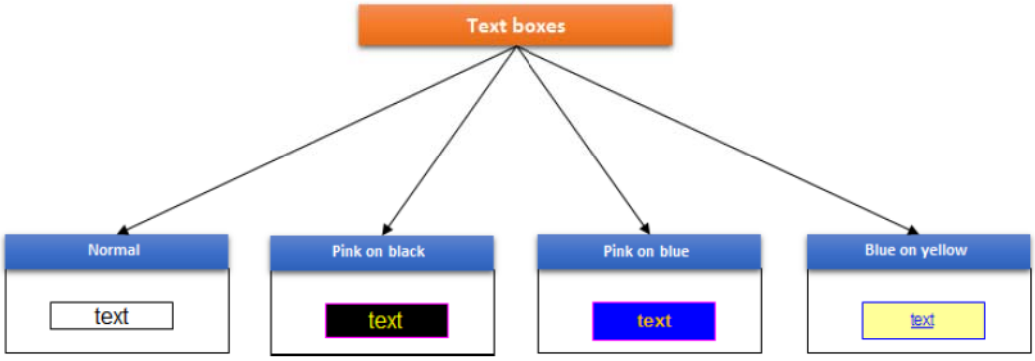


Figure 7: Textboxes representations

3.4.4 Date entry alternatives

Date entry constitutes a frequent task. Five alternatives were designed for date entry and are presented in Figure 8. The first three designs include the selection of day, month and year respectively using an "inflated" calendar control. The calendar control has embedded appropriate calculations based on leap years and number of days that each month includes, so that only the valid days are placed in the days' instantiation according to the selected month and year. This option is targeted to novice users, because of its simplicity and error prevention mechanism.

The fourth design is similar to the first three, with the difference that the selection of the user input is made using spinners. The user is able to press the up and down button to select the desired day, month and year accordingly. The spinners controls do not have inherent support for leap year calculations, hence the developer should take appropriate action during the instantiation of these controls in order to ensure that only the valid numbers will be displayed..

The last design contains drop down boxes with automated date enforcement. The user has to select year, month and day from the drop down boxes. The validation of the date takes place as the user changes his selection. This date input alternative is designed for expert users to improve speed and effectiveness.

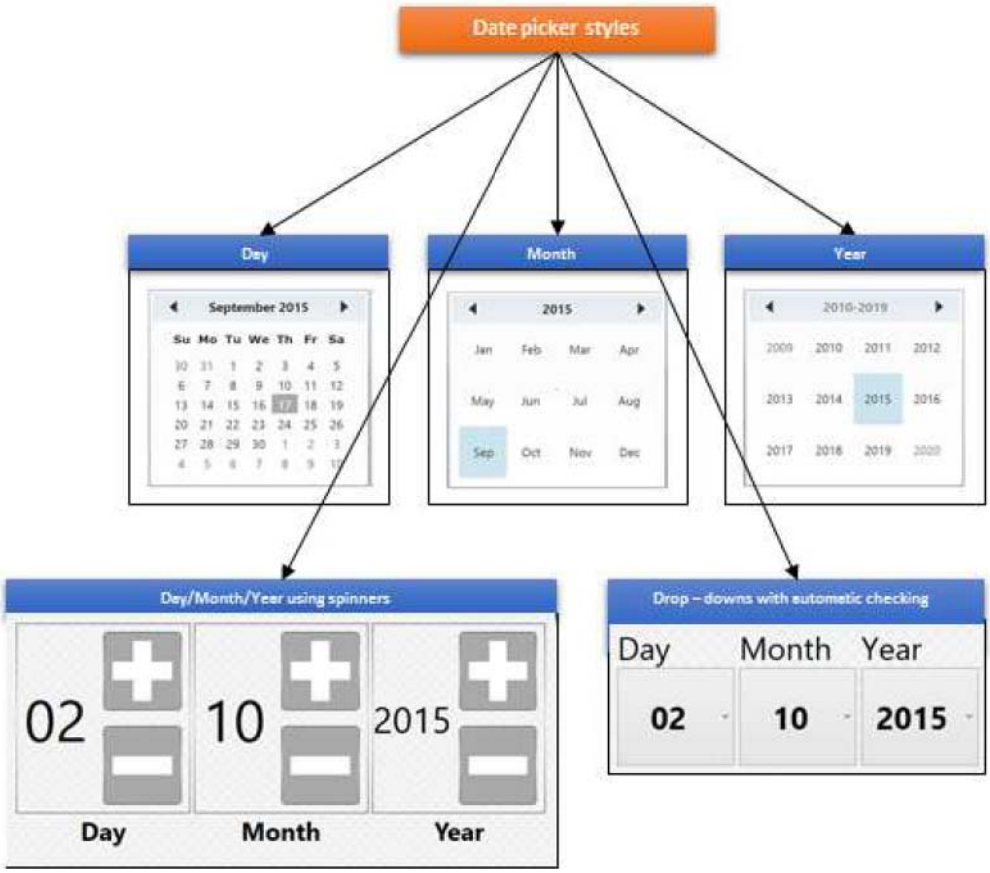


Figure 8: Date entry alternatives

3.4.5 Virtual keyboard entry alternatives

Text entry on touch devices can be a rather tricky task if it is not supported by appropriate virtual keyboards that can facilitate the process. Two alternatives virtual keyboards were designed to facilitate the text entry task on the robots onboard touch screen and are presented in Figure 9. The first design demonstrates a condensed keyboard that can be used by expert users for convenience and speed during the text entry task. The second design illustrates a bigger expanded keyboard with more margin between the virtual keys. This version has wider keys while the region where one key ends and another begins is marked with a “marching ants” line for better visibility in different coloring schemes. This version of the keyboard can be used for novice users or in situations where the user in not in a comfortable position in relation to the touch screen of the robot but needs to input some text.

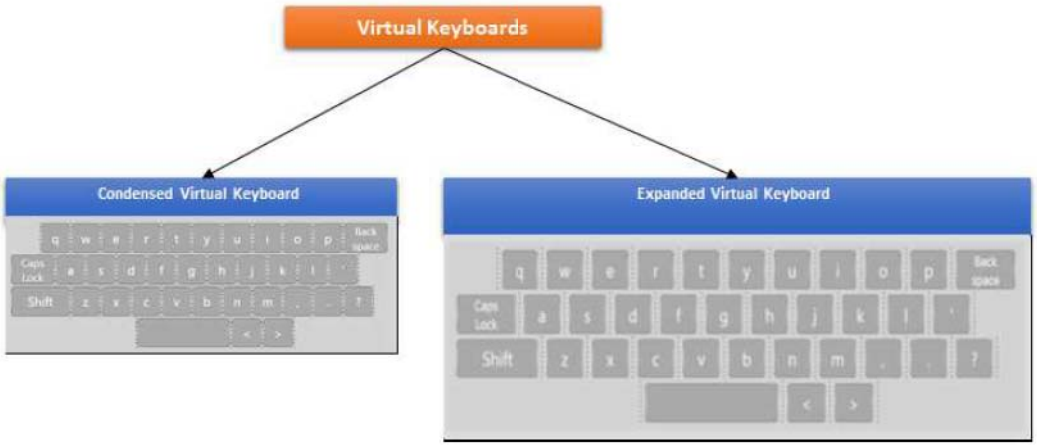


Figure 9: Virtual Keyboard alternatives

3.5 Generic robotic functional components in HRI design

3.5.1 Tablet PC

The tablet PC functional component is a touchscreen operated electronic device that is usually mounted on the main body of the robotic platform. It supports touch interaction and is responsible for hosting and displaying the different user interfaces of the robot's applications that implement the robot's functionality. An average tablet PC setup uses a relatively large touchscreen device e.g. 312w x 207d x 17h mm (used in the HOBBIT robot [136]) The tablet can be mounted or fixed by re-closeable fastenings on a small base plate which is mounted on the robot by tilt links so that its tilt angle is adjustable to fit the user's needs. If the base plate extends beyond the area of the platform, it is secured by a bumper. The optimal height to the lower edge of the touch screen has to been shown to be 72cm, which is the recommended height of an office desk for a seated user [137]. The height to the center of the touch screen must be within the range 76.0-106.5 cm, recommended for a touch screen for a seated user [138]. The angle of the touch screen is recommended to be 45 for a standing user when the center of the touch screen is below 104cm [138]. In cases where the robot is destined to be used more by sited users instead of standing users, the angle should be reduced accordingly. According to anthropometric data for British adults aged 65-80 years [137] the height from the floor to the elbow for a person sitting in a 45cm chair is 61.5-72.0cm (5%ile women - 95%ile men). This should be taken into account

in respect to the height of the center of the touch screen, in order not to cause discomfort for a seated user. The anthropometric data from the floor to the tip of the middle finger for a standing person is 55.0-69.5cm (5%ile women - 95%ile men). This should be also taken into account in respect to the height of the lower edge of the touch screen, so as not to cause discomfort for short usage for a standing user.

The touchscreen device is used as an alternative way of interaction, both as input (touch) and as an output device (screen). In terms of adaptation, the UI should be able to adapt both a priori (adaptability) and during the interaction process (adaptivity) in order to suit the preference and the dynamically changing needs of the users. For example the UI controls could increase in size, change colors and contrast or get replaced by simpler or more convenient ones if the user is not able to see properly or fails to accomplish the respective task/goal. The robot's touch screen will adapt according to the robot's distance from the user. For example when the user wants to initiate the "Clean floor" functionality, the robot must be in touch range for the user to be able to use the robot's touch screen. However when the actual operation of cleaning begins to take place and the robot moves away from the user then the touch screen becomes illegible. As a result there is no point in maintaining the actual state of the GUI. On the contrary the touch screen could be changed to illustrate a picture or drawing of the action in progress. In this example, the screen could change into showing a full screen "broom" which can depict the status of the robot at that given moment even if the robot would be far away from the user.

3.5.2 Hand(s)

This functional component refers to the potential hands of the robotic platform. The purpose of this component is to be used to grab things (e.g., from the floor, the table etc.), to physically manipulate switches or electric appliances (e.g. the robot can turn the lights on or off, or manipulate an inductive electric stove), or in terms of handover manipulation or other tasks that may require physical touch between the robotic platform and its users. All the above scenarios can be adapted to the preferences of the users (e.g. they don't want to be touched, they want to set a maximum allowed velocity for the moving parts of the robot, they don't like to participate in handover interactions etc.). Furthermore the hand can be used as an additional output modality

of the robot in order to give information to the user similar to how people use their hands while they talk. For example the robot could use its hand to point to things that concern the respective context of interaction, remind the user about the proper realization of hand gestures etc.

3.5.3 Wheels

This functional component refers to the wheels of the robotic platform. The wheels are used to move the robot around and fine-tune its position in respect to the user. Since robotic platforms can move around independently as opposed to other electronic appliances, this offers an important aspect of adaptation that should be taken into account. Hence, the capabilities of the moving robotic platform can be used to move closer to the user in cases of emergency where the user needs help, or when the user finds difficulty in reading the displayed information on the onboard screen. Finally the robot can be used to follow the user around the house, carrying objects for him/her while the speed of the platform can be adapted to the context of interaction and the preference of the users.

3.5.4 Speech Synthesis

The speech synthesis interaction modality is the functionality of the robot to produce synthetic speech in terms of auditory output or feedback to the users. It can be used either as a primary means of communication or as an additional modality used for redundancy purposes when the robot wants to communicate verbal information to the users or provide feedback. The potential adaptation that is supported by the framework for the speech synthesis modality include the selection of the gender of the output voice of the robot, its volume, rate, pitch as well as the vocabulary that is being used. For example, the voice could be selected to be male or female according to the current situation or preference of the user, the volume can adapt to the room's ambient noise or to the distance between the robot and the user making it louder or softer taking into account the cases where another resident could be annoyed by the volume increase.

3.5.5 Speech Recognition

The speech recognition input modality is responsible of recognizing and identifying a predefined set of vocal commands that the users can give verbally to the robot. This modality is used in terms of verbal control of the robot by the users, since the user is able of just telling the robot what he/she want it to do and the robot responds accordingly. The whole interaction is based on a predefined set of vocal commands that the robot is able to understand. The richer the set of commands is, the less the user is required to remember, since the robot understands more equivalent commands.

In terms of adaptation that is supported by the proposed framework in the context of the speech recognition modality, the robot can turn the modality on or off according to the respective level of ambient noise in the room or according to whether other modalities are concurrently active. For example the robot always temporarily turns off the speech recognition modality when it is providing auditory feedback to the user, as it doesn't want its own "speech" to be understood as commands to itself. Furthermore, a threshold can be provided for accepting or rejecting the recognition of verbal commands. In particular, since the recognition of each individual phrase or command can have lower of higher confidence levels, the speech recognition threshold can be defined in relation to the respective command(s). Furthermore, since the pronunciation of each user might differ, the recognition threshold could be different and thus has to be adapted to the particular user. Moreover, the recognition threshold can be further automatically adapted during the interaction process, either to further fine tune it to better understand each individual spoken commands, or to adapt to environment factors such as the noise of the room, or the distance between the user and the robot. Finally, similar sounding verbal commands can be analyzed in advance so that the individual recognition thresholds can be calculated and fine-tuned in advance.

In addition, the vocabulary that the user is able to use in order to give verbal commands to the robot can be adjusted to fit the cognitive levels of each individual user. Equivalent commands can be added to further relieve the user from having to remember specific commands. Furthermore the robot can react and respond

differently even to equivalent commands, since the context of each command can slightly differ. For example when the user asks the robot to show him/her the time, the robot can simply display the current time on the onboard screen, whereas if the user asks the robot the question "What time is it?" the robot can additionally respond verbally by telling the user the actual time! This leads to a more rich interaction experience, contributing to the users' subjective satisfaction, to building a more interesting relationship between the robot and the users and finally to a smoother learning curve and higher levels of acceptance of the robotic platforms.

3.5.6 Gesture Recognition

The gesture recognition input modality is responsible of tracking the user's skeleton and understanding a predefined set of gestures that the user is able to perform. This is a very important interaction modality since the user is able to communicate with the robot without using touch or speech. This can be extremely useful in cases where the user is unable to reach the touchscreen of the robot, or is unable to speak to the robot, for example in emergency scenarios where the user needs help. Snice the gesture are understood by the robot, it is capable of adjusting its position in order to better track and more accurately understand the user. The framework provides inherent support for the gesture recognition modality, while the robot is able to understand among a relatively easy to remember and perform gesture set. The framework provides continuous support for the recognition of an important subset of user gestures, such as gestures calling the robot for help, or instructing the robot to stop the current task. Furthermore, the individual subset of gestures that are to be used by the different applications can be easily configured. The individual gestures that are of interest can be loaded and unloaded by the respective applications at runtime by providing their identification name. Furthermore, a minimum recognition threshold for successfully understanding or rejecting a recognized gesture can be specified. Finally, the recognition engine can be fine-tuned to understand a specific subset of gestures for each user. This subset could be determined according to the success identification rate during the user training or it can be based on other factors such as the time of the day, when for example the user might be too tired to raise his hands up at night.

4 Architecture

This Chapter describes the architecture of the FIRMA framework. Section 4.1 discusses the low fidelity paper based architecture prototyping process that was performed in the early stages of the design. Section 4.2 discusses the high level functional architecture layers and describes the different types of functionality that they provide while section 4.3 discusses the orchestration of the conceptual layers that comprise the framework and describes their functional role.

4.1 Preliminary low fidelity paper based architecture prototyping

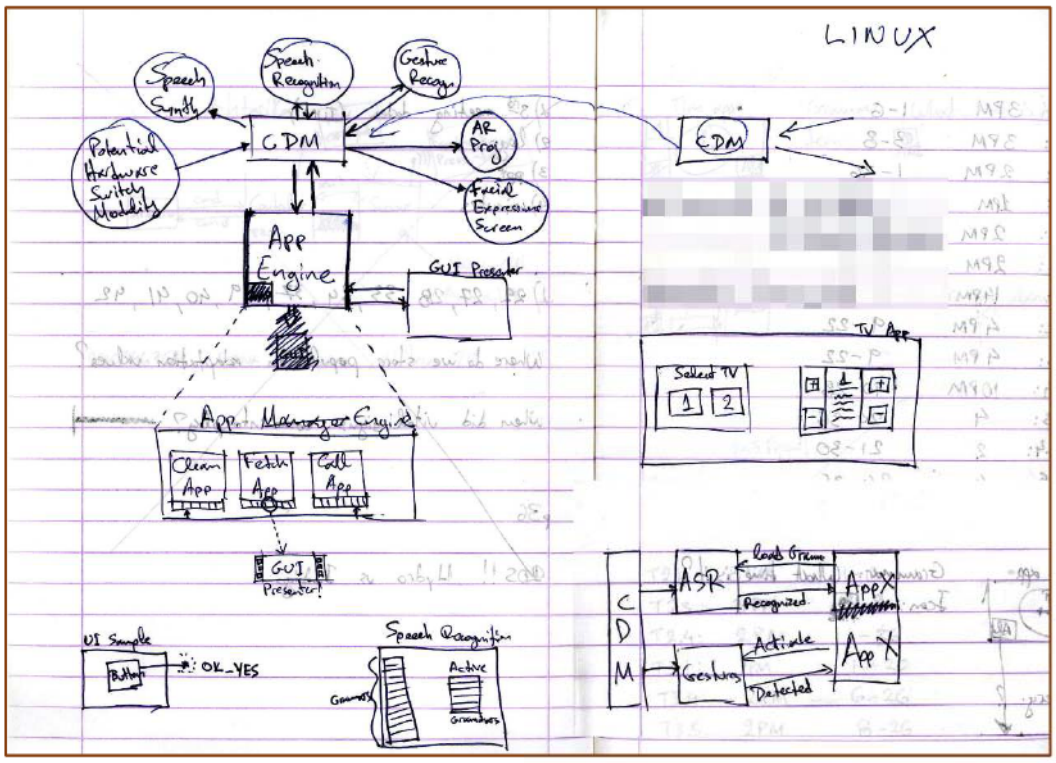


Figure 10: Low fidelity paper based architecture design for the FIRMA framework

In the context of a distributed ambient environment, system architectures become fuzzier mainly due to the fact that services distributed in the environment have their internal architecture and structure while the way that services are incorporated and

used forms another higher level architecture. A generic abstract view of the architecture employed by this research work is presented in Figure 10.

Figure 10 presents the preliminary low fidelity paper based prototyping regarding the proposed framework's architecture. The figure depicts the different interaction modalities that comprise the Communication Planner conceptual module of the FIRMA framework. In this module, the different modalities are being managed by the communication decision maker module which is responsible for selecting and fusing the appropriate modalities according to the preferences of the user and the context of use. The figure also demonstrates the relation between the different potential applications, the screens and dialogues that comprise them and the main GUI presenter window, namely the UI navigator window of the framework. Finally, the necessary bridge between the ROS backend and the framework is illustrated, while the communication between the modalities, the communication decision maker module and the different applications is outlined.

4.2 High level system architecture

The preliminary low fidelity paper based prototyping that is described in the previous section, led to the final designed of the FIRMA framework. The different general architectural layers can be seen in Figure 11.

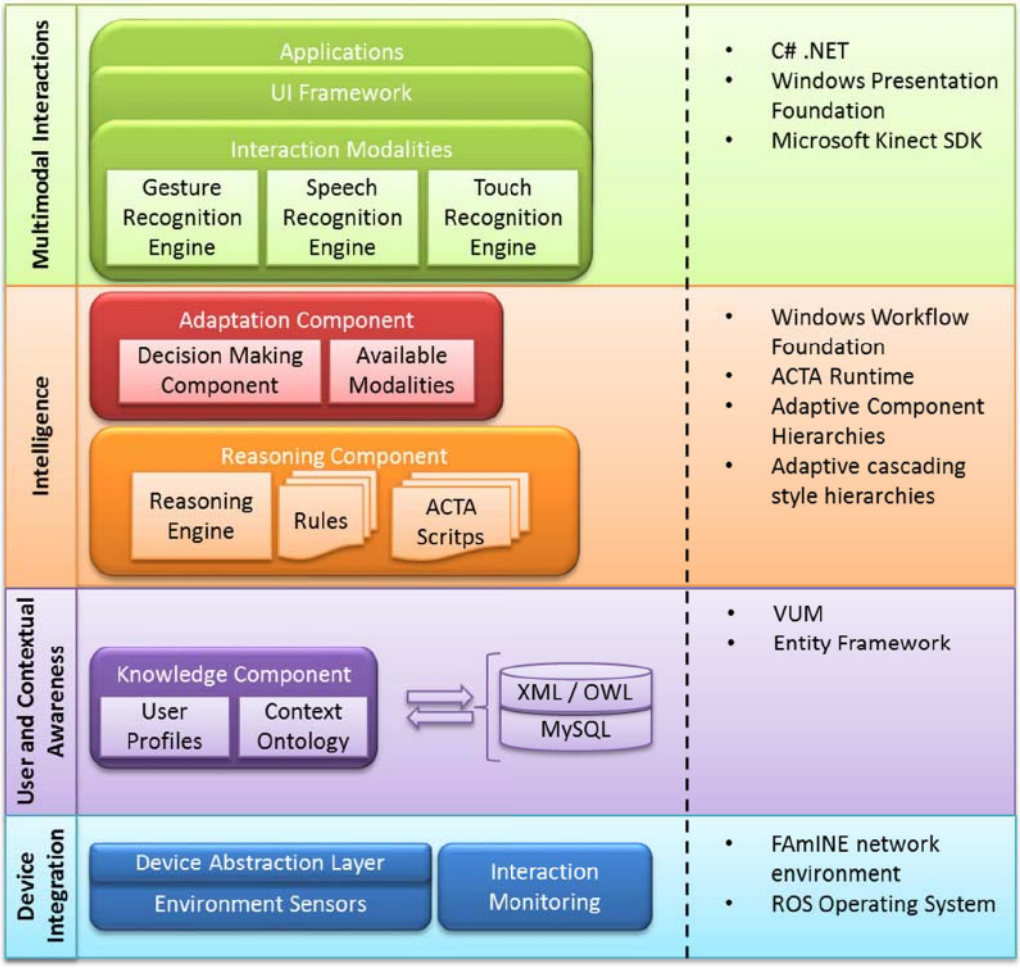


Figure 11: FIRMA framework's architectural layers

The FIRMA framework is comprised of several architectural layers, as depicted in Figure 11, that are described in detail in the following sections. In brief, the system is comprised of the following modules:

- Sensory infrastructure adapter modules that play the role of an intermediary transformation engine that undertakes the transformation of the information provided by the different environment sensors to pieces of information that can be understood by the system
- A user profiling and modeling subsystem that provides a priori knowledge used for the adaptation of the applications prior of user interaction

- An interaction monitoring subsystem that continuously monitors the ongoing interaction between the user and the system and provides the necessary input to the rules engine module
- A context information module that provides information about the context of use and the general context of the surrounding environment (this module uses the input from the sensory infrastructure adapter modules)
- A rules engine subsystem that is used at runtime, to infer facts about the active user who would is interacting with the respective application for adaptation purposes
- A decision making module that is used at runtime to activate or deactivate the necessary user interface components according to the user profiling and the ongoing interaction process
- A modality module for every different modality that is supported by the system (i.e., a gesture recognition module, a speech synthesis module, a speech recognition module, a touch gesture recognition module etc.)
- A set of user controls that support multimodal activation used for application development.
- A collection of adaptive component hierarchies mainly comprised by common abstract task hierarchies that are properly instantiated according to the user profile at runtime
- A collection of cascading style hierarchies that are used for adaptation purposes regarding the different framework elements and dialogues of the system, while contributing to a uniform look and feel throughout the system including the developed applications.

4.3 Orchestration of conceptual layers

Based on the aforementioned high level system architectural layers, the FIRMA framework comprises a collection of conceptual layers which can be seen in Figure 12.

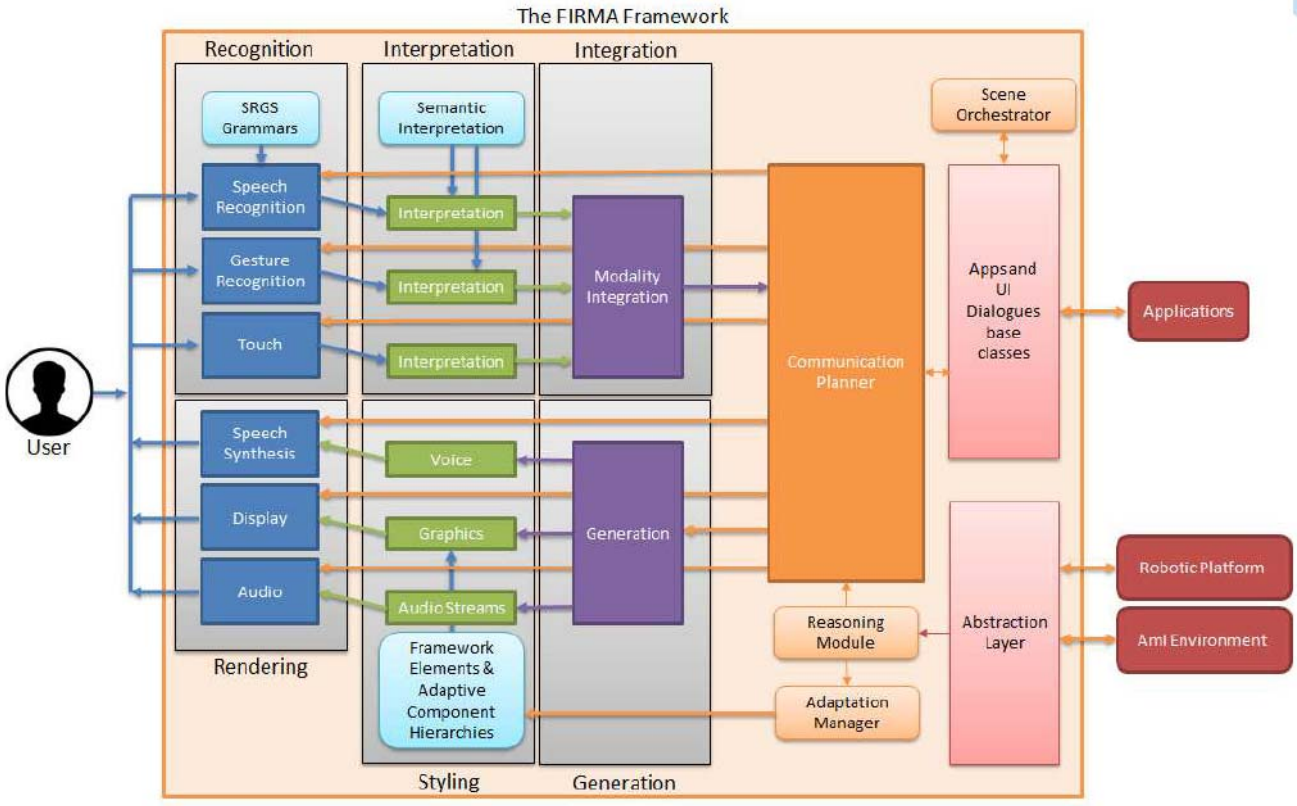


Figure 12: Orchestration of the different conceptual layers

4.3.1 The interaction recognition layer

The user is able to interact with the system through the interaction recognition layer. This layer consists of the different available interaction modalities that are provided. Additional interaction modalities can be added in future work such as hardware buttons and switches. The user is able to interact with the system using touch, gestures and voice. These modalities are adapted to the profile of the user, his preferences and the context of use. They are managed, selected and fused together by the communication planner functional submodule. The touch recognition modality corresponds to the touch interactions between the user and the robotic platform's onboard touch screen. The gesture recognition modality refers to the set of preselected gestures that the user is able to perform and the robot is able to understand and behave accordingly based on the context of the interaction. Finally, the speech recognition input modality refers to the predefined set of SRGS speech recognition grammars that describe the set of vocal commands that the robotic platform is able to understand. This set of SRGS grammars is loaded into the speech

recognition engine so that the robot will be able to interpret the user's speech accordingly.

4.3.2 The input interpretation layer

The output of the interaction recognition layer is fed into the input interpretation layer. This layer consists of the processing of the user input in term of semantic interpretation based on the context of interaction. Each input modality of the input recognition layer is interpreted accordingly to the profile of the user and the interaction context. The speech recognition modality is interpreted according to the semantic speech annotations that are included in the corresponding SRGS speech grammars. The gesture recognition modality input is interpreted according to the respective application's logic that is active during the interaction, as well as the context of the interaction. For example, the same affirmative gesture may have different interpretations according to the context that is being used and hence it could be interpreted either as a "YES" in the context of a question or as a "NEXT" in the context of an interaction process. Finally, the touch modality input can be interpreted based on the dialogue that is displayed on the respective moment that the interaction took place.

4.3.3 The integration layer and the low level framework architecture

The interpreted input is fed from the input interpretation layer into the modality integration layer, where the input from all the different available modalities is integrated based on high level integration scripting. For this purpose, the ACTA runtime (see section 5.1) is used to integrate all the available modalities into a uniform input channel that can be routed to the communication planner functional component in order to take the necessary decisions regarding the orchestration of the input and output modalities. The same integrated input becomes available to the respective active applications through the low level input mechanisms that the FIRMA framework provides through the base classes that the developed applications inherit. Furthermore, the different available applications can communicate with the scene orchestrator functional component in order to gain access to the functionality it

provides regarding the management of the different application screens and their display on the onboard robot screen. The input from the sensors of the robotic platform as well as the input from the ambient intelligence environment is transformed into system readable format. The input from these sources is then routed through the reasoning module of the framework in order to infer all the necessary adaptation and communication decisions. When there is need for output from the system to the user, the communication planner decides over the selection and the fusion among the different available modalities to generate the information that is going to be conveyed to the user.

4.3.4 The output styling layer

When there is information that is needed to be conveyed to the user, the information goes from the generation layer to the output styling layer. This layer is where all the styling over the information delivery takes place. For each one of the available modalities, the appropriate styling is selected according to the user model, his preferences and the context of the interaction. The styling can refer to the output for the speech synthesis modality, to the output for the UI display modality or the output for the audio modality. For the speech synthesis modality, the appropriate voice is selected according to the preferences of the user. Additionally, the appropriate rate, volume and pitch of the voice is selected and the output speech is styled using the SSML markup language. For the UI display modality that corresponds to the touch display onboard the robotic platform, the appropriate UI selection and adaptation takes place according to the decisions of the adaptation manages functional component. The appropriate UI elements and dialogues are selected, the appropriate component hierarchies are instantiated and the output is delivered to the robot's display for the user to interact with. Furthermore, for the audio output modality, the appropriate auditory feedback is selected and the parameters of the audio output are specified. Finally, the output from the output styling layer is wired to the output rendering layer.

4.3.5 The output rendering layer

The final stage of the output delivery is the output rendering layer. This is the layer responsible for delivering the actual output to the users. It comprises of the different available output modalities as they have been selected and fused by the communication planner functional component. For the speech synthesis output modality, the actual speech is generated based on the SSML annotations from the styling layer and the final auditory feedback is delivered to the user. For the touch display output modality, the appropriately selected framework elements, components and dialogues are instantiated and the result is presented on the robot's onboard display. Finally, for the audio output modality, the appropriate adaptation parameters are applied and the auditory feedback is delivered to the user.

5 Implementation

This chapter presents the implementation of the FIRMA framework. The result is a fully integrated development framework that can support the design and development of elderly friendly, multimodal interactive applications that are deployed on domestic robotic platforms which can in turn take full advantage of the possibilities offered by AmI environments. The results of this research effort include all the necessary tools and building blocks for the creation of speech enabled, voice recognition enabled, gesture recognition enabled, and touch enabled adaptable and adaptive interactive applications.

5.1 ACTA: A general purpose finite state machine (FSM) description language for ACTivity Analysis

ACTA is a general purpose finite state machine (FSM) description language [139]. ACTA's primary design goal was to facilitate the activity analysis process during smart game design by early intervention professionals who are not familiar with traditional programming languages. However, developers can use ACTA not only for developing event-driven sequential logic games, but also for applications whose behavior is composed of a finite number of states, transitions between those states and actions, as well as for application based on rules driven workflows.

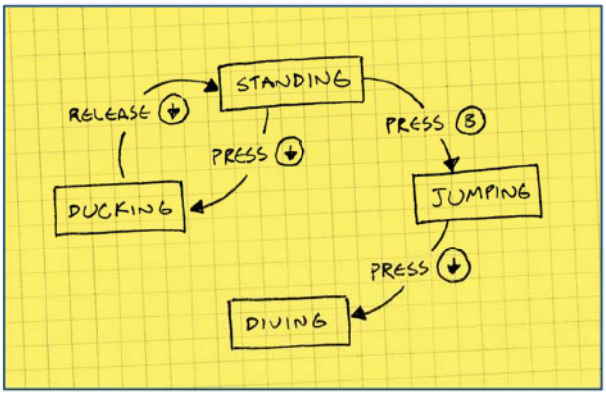


Figure 13: A Finite State Machine example[8]

[8] Image taken from http://www.padaonegames.com/bb/

A finite state machine consists of states, inputs, and transitions. A script in ACTA can describe a finite state machine as depicted in Figure 13, which illustrates a flowchart of a hero game. The flowchart consists of boxes for each action a hero can do such as standing, jumping, ducking, and diving. When acting, the hero can respond to a button press and perform a different action. Furthermore, in each of the depicted boxes, there are arrows connecting the current action with a next one according to the labeled pressed button. In detail, the ACTA script of this example consists of the following parts: a) a fixed set of states that the machine can be in (e.g., standing, jumping, ducking, and diving), b) current state, the machine can only be in one state at a time (a heroine can't be jumping and standing simultaneously), c) a sequence of inputs or events (e.g., the raw button presses and releases), and d) a set of transitions from current state to a new one based on incoming input or event. When an input or event comes in, if it matches a transition for the current state, the machine changes to the state that transition points to. An example of the hero game in ACTA script is depicted in Figure 14.

```
1
2   State "STANDING"
3   {
4       when (PressedButton == "ARROWDOWN") { NextState = "DUCKING"; }
5       when (PressedButton == "B") { NextState = "JUMPING"; }
6   }
7
8   State "DUCKING"
9   {
10      when (ReleasedButton == "ARROWDOWN") { NextState = "STANDING"; }
11  }
12
13  State "JUMPING"
14  {
15      when (PressedButton == "ARROWDOWN") { NextState = "DIVING"; }
16  }
17
```

Figure 14: Finite states of an action game in ACTA script.

Authors can use ACTA for scripting a smart game or application in general, by outlining a set of finite states and event-driven transitions (i.e., a transition from one state to another is triggered by an event or a message). Thus, ACTA facilitates event driven programming, in which the flow of the program is determined by events, event handlers, and asynchronous programming.

ACTA support the dynamic invocation of functions through reflection. The function calls that are contained in the ACTA script are searched at runtime when the script

instructs for a function call. The ACTA backend starts searching for the desired function in the base classes of the framework. Upon potential failure the search continues in the developer created derived classes. Polymorphism is inherently supported by the backend as it is written in C# which natively supports virtual functions. The invocation of the functions is done through reflection and as such, the appropriate derived class' function gets called.

Property getting and setting, is done through auxiliary functions namely GetProperty(string name, object value) and SetProperty(string name, object value) which operate on the application's derived class instance. This approach leverages the properties info function calls that use reflection to accomplish their goal similarly to the invocation of real functions.

5.2 ARMA: Extending ACTA Runtime to support the development of Multimodal elderly friendly Applications

ACTA's runtime has been adopted and adapted to fit the needs of the creation of multimodal interactive applications. ACTA's runtime is based on the Windows Workflow Foundation framework. The ACTA IDE is used to code all the application interaction logic which can then be extracted to an XML rules file for further use. The rules file can be loaded into a Windows Workflow Foundation Rule Engine which is an event driven reasoning engine that can run the provided rules and conclude to the desired actions, results and transitions between the different states that describe the application logic.

The main workflow for creating an interactive application includes the definition and design of the different application screens and then the definition of the different application states. Usually, one state is then mapped to one application dialogue screen. However, states with no visual output can exist, as well as multiple states can correspond to the same UI dialogue being displayed at that time. The integration of the different modalities is discussed below.

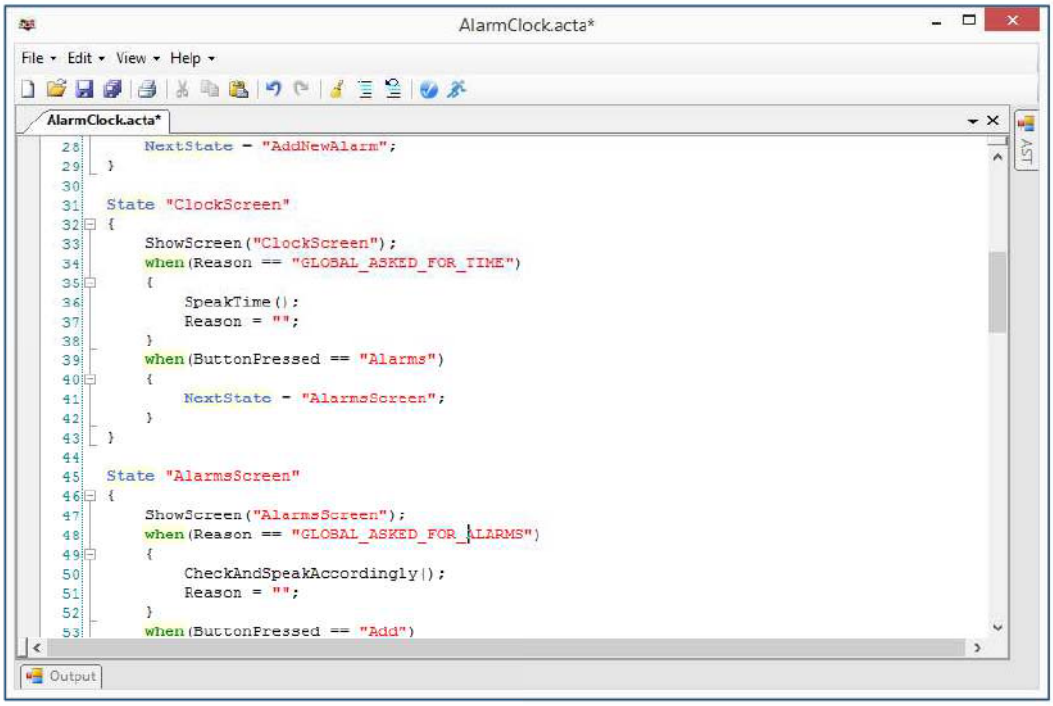


Figure 15: AlarmClockApplication ACTA logic snippet

Figure 15 shows a fragment of the ACTA script used to encode the logic behind an Alarm Clock Application. The ACTA's runtime part that had to do with maintaining and managing the different states of an application or game has been rewritten to support elements such as the current state the program is in, the previous state for potential navigation to earlier states, the next state that the application is desired to go to upon reception of specific input and a return state where the application logic can return after following pre-configured framework UI hierarchies. To this end, ACTA uses 3 fundamental keywords for scripting a finite state machine: a) State, b) NextState, c) PreviousState and d) ReturnState.

A finite state is described using the keyword "State" that defines a state statement. A state statement defines a block that contains: a) optional macros for execution when entering the state, and b) transitions. The latter may be either an explicit state change using keyword NextState (e.g., NextState = "state's id") or a set of when statements in order the machine to remain in the current state until the condition of a when statement becomes true and fires its transition

The NextState keyword is used to set a new state. Using an assignment expression, the author simply sets the id of the next state to the reserved word NextState (e.g., NextState = “state id”). Event-driven transitions are described using the keyword when. This keyword is used to define a when statement. In other words, when a condition becomes true, a transition from the current state to a new one is fired.

The PreviousState keyword is used to hold the previous state an application has been into. This is useful for transitions when the user initiates a multi-dialogue process and wants to be able to navigate back and forth, or more importantly when the natural UI dialogues sequence is interrupted due to a higher priority event. Using the PreviousState keyword the developer is able to instruct the application to return to its previous state once this higher priority event is handled.

Moreover, the ReturnState keyword can be used alone or in conjuction with the PreviousState keyword to accomplish more complex navigation and transition scenarios where adaptive component hierarchies are included in the state traversal or when higher priority events include more than one displayed dialogues which in turn implies more than one States being traversed.

```
State "AlarmElapsed"
{
    ShowScreen("AlarmElapsed");
    ReturnState = PreviousState;
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        DeleteAlarm();
        NextState = PreviousState;
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "AlarmSnoozed";
    }
}

State "AlarmSnoozed"
{
    ShowScreen("AlarmSnoozed");
    SnoozeAlarm();
    when(Time == 5)//or when(ButtonPressed == "OK")
    {
        NextState = ReturnState;
    }
}
```

Figure 16: AlarmClockApplication ACTA logic snippet

For example, in the above snippet the ReturnState keyword is used to store the PreviousState keyword value when an Alarm goes off. This way, in the case where the user decides to acknowledge the elapsed alarm, the program can go back to the previous state, whereas if the user decides to snooze the elapsed alarm, the program can navigate to a different dialogue where the user is informed about the alarm having been snoozed and then return to the "ReturnState" state after five seconds.

Finally, states such as the SuccessState and AbortState have been introduced to support the implementation of adaptive component hierarchies, whose integration demanded distinguishing between completed and aborted tasks.

5.2.1 Loading and unloading rules at runtime

A very useful functionality that has been added to the ACTA backend is the option to load and unload rules at runtime. This contributes to the reduction of the rules that are loaded at any given time. Furthermore, it offers the ability to change the behavior of the developed applications based on the subset of rules that are loaded at a given point in time. In other words, the developed programs are able to load extra rules at runtime in order to alter their functionality.

This enables the usage of abstract task hierarchies that can be instantiated at runtime, while the respective rules that support their functionality are loaded at runtime. Furthermore, this addition opens new paths for adaptation based on the extra subset of loaded rules. For example, the experience of the user can be taken into account when he/she is expected to fulfill specific tasks and the UI that he/she is presented with can change accordingly. Moreover, tasks that are frequently required are automatically adapted and embedded into the framework such as the task for the selection of time.

The dynamic loading and unloading of rules has been implemented in full compliance with the functionalities of the language for rule activation/deactivation, rule prioritization etc. The backend has been extended to support the dynamic rule loading by respecting the aforementioned properties and treating them appropriately. Since

the Microsoft Workflow Foundation does not provide the necessary functionality for merging rulesets, the whole process of the dynamic rule loading had to be added. Loading and unloading extra rules as needed, is a lot different than having all the rules loaded at all times and then activate or deactivate a subset of them as desired. The latter approach can have a huge performance impact in the whole rule engine and was thus avoided. During the loading of new rules, the rule engine is temporarily paused and the new rules are appended to the currently active ruleset. The old pre-existing rules are not removed or disabled because their functionality is still needed as the new rules do not substitute the old ones but merely temporarily extend the functionality of the application. After loading the new rules, the back-end ACTA data structures are augmented accordingly to support the rule addition without affecting the language's mechanisms such as the mechanism for dynamic rule activation and deactivation or the capability for rule prioritization. Upon a successful append, the rule engine is resumed to activate the functionality that is offered by the new rules. Finally, when the functionality that is offered by the new rules is no longer needed, they are unloaded and the back-end data structure changes are reverted.

When a new set of rules is loaded, it is validated against the rule engine and then run against the instance of the application. The validation is always successful because all the function calling and property manipulation of the rules is implemented through a set of auxiliary helping functions namely the *DoAction*, the *SetProperty* and the *GetProperty* accessor function. This functionality is inherently embedded into the ACTA language so that the produced ruleset is transcribed into using these functions. This approach has the advantage that most of the fatal conditions can be silently ignored with the corresponding error messages being printed on an error log file while the state of the application remains stable. This means that if the ACTA script contains instructions for calling functions or setting properties that can't be found neither in the framework base classes nor in the developer-created derived instances, the invocation of those functions can fail silently without compromising the stability of the whole system.

5.2.2 Modality integration

Modality integration has been realized by leveraging the different modality generated events and consolidating them at a higher level where the corresponding application can treat them appropriately. This was achieved by implementing various mechanisms in the ACTA backend and in the frameworks base classes.

The framework contains backing fields for modality events such as ButtonPressed, SpeechRecognized and GestureRecognized. The ACTA backend was extended accordingly to support these fields. When the user interacts with the UI using touch events and touch gestures, these interactions are interpreted into the corresponding events and transferred to a higher level inside the application. For example, when the user presses a button, it generated the ButtonPress event in the base class of the application which is part of the framework. The base class contains the rule engine that can run the loaded ruleset against such events. The user is then able to interact with the UI based on the functionality that has been coded into the application's ACTA script. The result of the activation of the different rules includes state changes and UI dialogues activation in the derived application classes. This way the sequence of the application's dialogues can be easily tweaked and rearranged by the developer as needed.

A very useful feature of the FIRMA framework is the functionality it provides for modality integration at two different levels. The various available modalities can be integrated in the scope of an application's dialogue screen where the developer has to cater for each of the available modalities' events and act accordingly. Another approach would be the consolidation of the modalities into a single one and then develop a corresponding modality handling script that caters for this consolidated modality. Furthermore, modality consolidation can happen either in the scope of an application's dialogues or in the higher scope of application ACTA logic. For example, if the user can press a button by touch, voice or gesture, the different modalities could be consolidated into the button press in the scope of the application dialogue or in the ACTA scripting logic scope which is at a higher level. The developer then would only need to cater for the single touch press modality as the other two would automatically get consolidated into the touch modality scope.

Taking the modality events and raising to a higher level where they can be easily handled by the ACTA script contributes to the modular nature of the proposed framework's architecture as the framework's components are loosely coupled and completely asynchronous. Additional modalities such as hardware switches and different kinds of sensors and actuators can be incorporated into the framework with minimal effort, extending the provided functionalities and conforming to the user's needs.

5.2.3 Dynamic rule activation to improve performance and prevent instantaneous hopping among different states

While trying to incorporate all the different modalities and execute the desired action from the ACTA script at a higher level, an interesting program behavior appeared. There were apparent cases where a change in a property of the application's base class led to multiple instantaneous state hopping with adverse results. The issue was further investigated and discovered to occur for the "State" property as well as for any of the aforementioned modality back-end properties. This proved to be quite interesting as these were the properties that were used the most in the ACTA script which in turn was an expected occurrence, since these properties drive the application logic. Additional code inspection and retrospection revealed the causes of this behavior.

The hopping behavior that is discussed above was found to be caused by the way that ACTA handles the activation and deactivation of rules at any given time. In order to support the dynamic activation and deactivation of rules which is an inherent ACTA functionality, the backend namely the instantiation and the management of the rule engine has to take extra care of the initial status of each rule. When the rules are first loaded into the rule engine, the engine creates an index that stores the initial status of each loaded rule which can be either "active" or "inactive". Later on, this information is taken into account in order to minimize the number of concurrently active rules during an instance run of the engine which in turn results into the increase of the engine's performance as it reduces the time needed to reach to a result.

The activation of the engine is fully asynchronous and event driven. The backend holds a list of all the available agents, namely the class instances that need rules to be run against them in order to support reasoning functionalities. When a class needs to use the engine's functionality, it registers itself to the backend through conforming to the *PropertyChanged* component model. This model dictates that whenever a property of this class changes, the class is obliged to emit a PropertyChanged event which contains the name of the property that has been changed. Whoever is interested into these event changes, has to register to the EventHandlers of these events in order to be notified accordingly. In this scope, the classes that want to support the reasoning functionalities of the engine register themselves to the backend while the engine starts listening for any changes in the classes properties. When a property of a registered class changes, the engines detects the emission of the respective signal and triggers an activation of the necessary rules over the instance of the class. This is an asynchronous process that uses a concurrent queue to store the different changing properties, making the consequent property changes as quick as possible. If the change of the properties was a blocking process that had to wait for the engine to complete each run, it would have catastrophic impact on the performance of the system even though each run requires only a few milliseconds. This could easily be observed when multiple properties were required to change over a single iteration.

Combining together the asynchronous nature of the engine runs, with the offline registration of the initial state of each rule which was discussed above, pointed out the origin of the multi-stage hopping behavior. Before each activation of the rule engine, the system has to decide which rules are capable of affecting the state of the class. In other words, the system is capable of filtering the rules that are involved in each activation of the engine, based on their content. For example, whenever the property "State" changes, it would be reasonable to only activate rules that have the property "State" in their activation conditions list. The engine was able to filter out rules that did not contain the changed properties in their activation conditions list, which vastly increased the performance of the system, especially in cases where the logic of the application was complex and required a lot of rules.

The order in which the different rules were declared, played an important role in the aforementioned adverse behavior as well. A specific example was discovered where there were three rules that contained the property “State” in their activation conditions list. Each rule was dependent on a different state of the application, but all three of them had at least one transition that was caused by at least one modality. In this specific example, the common modality was the press of a context-positive button (See Figure 17).

```
State "state1"
{
    ShowScreen("screen1");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        NextState = "state2";
    }
}
State "state2"
{
    ShowScreen("screen2");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        NextState = "state3";
    }
}
State "state3"
{
    ShowScreen("screen3");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        InteractionCompleted();
        NextState = ReturnState;
    }
}
```

Figure 17: Problematic ACTA snippet, leading to stage hopping

The above script snippet results into several rules, three of which have a logical conjunction in their activations list that mentions both the property “State” and the property “ButtonPressed”. The first rule required both that the state should be “state1” and that the button should be pressed in order to trigger a transition to the state “state2”. The second rule required that the state should be “state2” and similarly triggered a transition to the state “state3” when the button was pressed. Finally the third rule required that the state should be “state3” and it triggered a transition to an “Exit” state when the button was pressed. The order of the declaration of the above

rules in combination with the implemented backend functionality led to the state hopping behavior. Whenever the application was in “state1” and the context-possitive button was pressed, a run of the rule engine was triggered. All three rules were activated during that run because all three of them contained the “State” property into their activations lists. As a result, when the first rule was checked for pertinence it was found to be applicable because the application was in “state1” and the button had just been pressed. The activation of the first rule resulted into the application being forced to transit to “state2”. Subsequently, when the second rule was checked for applicability, it was found to be compliant because the application was already in “state2” and the button had just been pressed because the engine was still in the same run instance. As a result the application was instructed to transition to “state3” which would be the reason for the activation of the third rule when it was checked for applicability, resulting in the application being instructed to transition to the “ReturnState”. The above workflow resulted into the application erroneously transitioning between three different states during a single engine run cycle while functionally skipping the respecting UI dialogues.

This proved to be a logical error in the activation logic of the different rules during the pre-screening process, right before the engine activation. It was apparent that only the first of the three rules should be active during the first run of the engine because it is the only rule that depends on the state of the application being “state1”. This was extended to apply to all the other available modalities as well, and the above undesired behavior was eliminated.

Similarly to the above, a few necessary auxiliary helper functions were added to the engine’s backend to reset “used” properties after they had been involved into an engine run cycle. For example, if an engine run cycle was triggered by the “ButtonPressed” property, the property should be reset after that run cycle was completed to avoid undetermined behavior in consequent run cycles caused by rules that are based on that property.

5.3 Interaction modalities

Designing and developing multimodal application for the elderly user group can be very challenging due to the vast heterogeneity that characterizes it. Different users are facing different difficulties in their everyday lives depending on their level of cognitive and psychosocial functioning decline due to aging. The HRI (Human-Robot Interaction) field can significantly help the users of this group, by adapting and personalizing the functionalities and the behavior of household assistive robots to fit the needs of their owners. Such needs are differentiated based on both the abilities or disabilities and the preferences of the individual persons.

The modalities that have been developed and integrated into the proposed framework range from speech recognition and synthesis to gesture recognition and touch interactions. They all have been developed to be fully extensible and configurable both at startup and at runtime so that they can change to reflect the changing needs of the users or the dynamically changing factors of the surrounding environment e.g. ambient lighting, environment noise, active electric appliances etc. In addition, the configurable parts of the developed integrated modalities have been offered as ROS services to the system, to support dynamic adaptation based on interaction logic that runs outside of the tablet PC of the robot.

The developed interaction modalities are discussed in the following sections.

5.3.1 Speech Recognition modality

Speech is an effective and natural way for people to interact with applications, complementing or even replacing the use of mice, keyboards, controllers, and gestures. A hands-free, yet accurate way to communicate with applications, speech lets people be productive and stay informed in a variety of situations where other interfaces will not.

Speech recognition allows users to interact with and control speech-enabled applications by speaking. The speech recognition modality has been included in the available interaction modalities of the framework that is discussed in this thesis. The users are given the opportunity and the possibility to interact with the robotic

platform using their voice, in the form of simple vocal commands. This is a set of pre-agreed verbal instructions that can be recognized and understood by the system, encoded into the form of speech recognition grammars as discussed below. The developed framework offers acquisition and monitor of speech input, support of speech recognition grammars that produce both literal and semantic recognition results, capturing information from events generated by the speech recognition, and full configuration and management of the parameters of the provided speech recognition engine.

The speech recognition engine that has been developed to cover the speech input needs of the proposed framework is based on Microsoft Speech Platform v11.0 and is fully configurable. The Microsoft Speech Platform is a product from Microsoft designed to allow the authoring and deployment of IVR applications incorporating Speech Recognition, Speech Synthesis and DTMF. The Microsoft Speech Platform SDK provides a comprehensive set of development tools for managing the Speech Platform Runtime in voice-enabled applications, including the ability to recognize spoken words (speech recognition). Also included in the SDK, the Microsoft Grammar Development Tools provide a comprehensive set of command-line applications that can be used to validate, analyze and tune grammars for speech recognition.

A speech recognition grammar is a set of word patterns, and tells a speech recognition system what to expect a human to say. For instance, in an auto telephone call responder scenario, the user would be prompted for the name of a person (with the expectation that his call would be transferred to that person's phone). The respective application would then start up a speech recognizer, giving it a speech recognition grammar. This grammar would contain the names of the people in the application's directory and a collection of sentence patterns that are the typical responses from callers to the prompt. Microsoft's Speech Platform can load and recognize speech grammars written using SRGS-compliant XML markup. Speech Recognition Grammar Specification (SRGS) is a W3C standard for how speech recognition grammars are specified. An example SRGS grammar that was used in the Alarm Clock Application use case can be found in Appendix II.

The following points present the capabilities, configurability and adaptability of the developed speech recognition module:

- The vocabulary and the variety of the individual equivalent commands that can be used by the users and understood by the system can be tweaked to fit the needs of the different users based on individual speaking habits. This means that the users are able to interact with a customized pre-agreed set of vocal commands based on their most frequent or everyday selection of words. The design and development of individual grammars can be done offline in cooperation with the end users who can select the sets of recognized spoken words and commands individually.
- The recognition results can contain both literal and semantic annotations (e.g., command: Turn Off Light, semantics: Bedroom) which gives the possibility of supporting different robot responds and reactions for different commands, even if they are equivalent. In another example, the user could ask the robot to either display the user's daily calendar on the robot's onboard screen by saying "Show me my appointments", or ask the robot "What's on schedule for today". In the latter case, the robot can treat the request differently and assume that the user expects some kind of response in addition to the displayed information and hence provide additional auditory feedback by "reading" the daily schedule to the user. Such differences in recognized commands can be semantically annotated and treated differently but the developed application's logic. To this end, the framework supports this annotation mechanism by providing dedicated back-end annotation fields which can hold the reason behind the robot's actions at any given time. This is discussed in more detail in section 5.5 in this Chapter.

- Furthermore, the speech recognition confidence threshold is able to be configured and adapted individually for each user based on the selection of spoken commands, the pronunciation clarity of the user's speech as well as the level of ambient noise and the distance between the robot and the user. This means that the higher the potential confusability factor of the selected phrases, the higher the recognition cutoff threshold should be. Furthermore,

the recognition threshold should be lowered as the user and the robot moves away from each other, or increased in cases of higher ambient noise. In this scope, the developed speech recognition engine can provide useful information regarding the quality of the input audio signal by providing insights about potential problems when the audio input has too much background noise, when it is too loud, too quiet, too fast or too slow, or when no audio input signal is detected (which most probably refers to hardware failures). Furthermore, the engine can differentiate between receiving silence or non-speech background noise and receiving speech input so that the Communication Planner submodule (discussed in 5.8.2) can turn the speech recognition engine on and off. Figure 18 shows a Speech Recognition instance performing speech recognition, where the ASR engine has successfully recognized a literal phrase with a confidence level above the predefined threshold. The recognition confidence is around 55%. The recognized phrase has attached semantic annotations, while the ASR engine has informed the framework that the audio signal had been too soft, hence the relatively low recognition confidence. Finally the engine instance has provided an update on its status, including a list of currently loaded grammars as well as their current state (active or inactive).

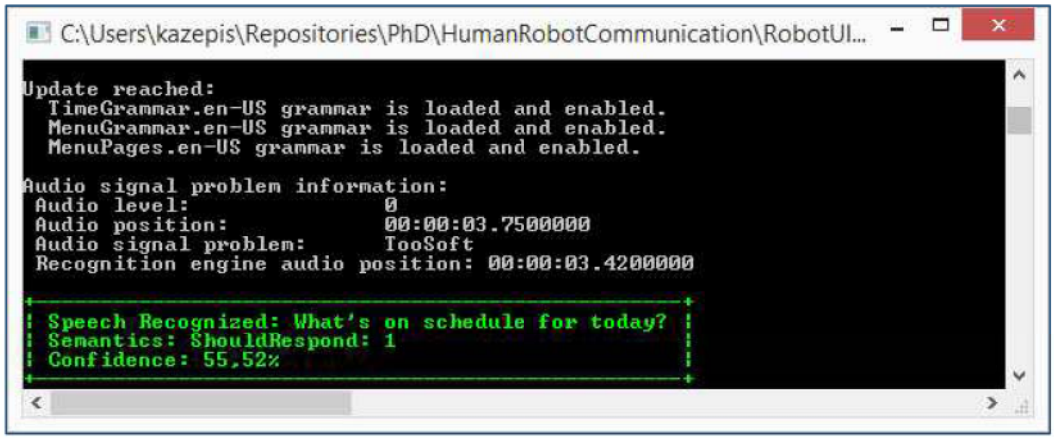


Figure 18: Speech Recognition instance showing the recognition of a literal phrase along with attached semantic annotations, potential audio signal problems and engine status regarding the set of loaded speech recognition grammars.

- In addition, voice prompts that can be easily confused or cause false positives can be easily identified and preferably avoided. The Microsoft Speech Platform provides a tool called Confusability that can run on user defined grammars, analyze them and recognize phrases that could potentially be misrecognized for other phrases in the grammar(s). That is, the tool detects the risk that one phrase may be falsely accepted for another phrase. The tool may detect the following types of confusable phrases:
    - Homonyms (for example “fly it” and “flight”; “bare” and “bear”)
    - Near homonyms or rhyming phrases (for example “bag it” and “tag it”)
    - Unintuitive confusable phrases that the recognizer identifies as having overlapping acoustic similarities (for example “stop” and “up”)

  The tool provides a metric that quantifies the degree of these risks. The output of the tool can be used by the designer of the specified grammar to tweak it in order to avoid potential misinterpretation of ambiguous utterances.

- The provided speech recognition engine instance loads and compiles all SRGS grammars that it finds in a configurable folder at startup. Compiling XML grammar to a binary grammar file with the .cfg extension can reduce the time consumed by searches for a match, especially in grammars that require recognition of a large number of words and phrases.
- Compiled grammars can be loaded and unloaded at runtime to reflect the needs for speech recognition for the currently running application at any given time.
- The startup and shutdown or the temporary disablement of the speech recognition engine is the responsibility of the Communication Planner as discussed in 5.8.2. For the rest of the developed applications, the usage of the speech recognition engine is transparent. The framework provides the necessary mechanisms for mapping the various provided user controls with the respective SRGS grammars that have been developed for this purpose as discussed in 5.2.2. The applications that have been developed based on the proposed framework can access the speech recognition engine at any given time through the Singleton instance design pattern.

- In addition, the functionality for runtime grammar manipulation is made available through a ROS service to the rest of the robot's applications that don't run on the tablet PC.
- Finally, the implemented speech recognition engine complies with the IDisposable design pattern which is implemented to ensure the proper disposition of the managed resources of the engine, namely the speech recognizer object.

5.3.2 Speech Synthesis modality

The most common way for human interaction to understand each other is through communication with each other, so as in the human robot interaction. The ability to talk is one of the most important technologies in the field of intelligent robotics. When it comes to talk there are two basic type of signal transmitted by people: Auditory and Visual. Speech synthesis is a crucial ability for auditory signal. Speech synthesis is the computer-generated simulation of human speech. It is used to translate written information into aural information where it is more convenient, especially for mobile applications such as voice-enabled e-mail and unified messaging. It is also used to assist the vision-impaired so that, for example, the contents of a display screen can be automatically read aloud to a blind user. Speech synthesis is the counterpart of speech or voice recognition.

The speech synthesis modality of the proposed framework has been based on the speech engine functionality provided by the *Microsoft.SpeechSynthesis* namespace. This namespace contains classes that offer the initialization and configuration of a speech synthesis engine, the creation of prompts, the generation of speech, and the modification of the synthesized voice characteristics. Speech synthesis is often referred to as text-to-speech or TTS.

A speech synthesizer takes text as input and produces an audio stream as output. A synthesizer must perform substantial analysis and processing to accurately convert a string of characters into an audio stream that sounds just as the words would be spoken. The easiest way to imagine how this works is to picture the front end and back end of a two-part system.

The front end specializes in the analysis of text using natural language rules. It analyzes a string of characters to determine where the words are (which is easy to do in English, but not as easy in languages such as Chinese and Japanese). This front end also figures out grammatical details like functions and parts of speech. For instance, which words are proper nouns, numbers, and so forth; where sentences begin and end; whether a phrase is a question or a statement; and whether a statement is past, present, or future tense.

All of these elements are critical to the selection of appropriate pronunciations and intonations for words, phrases, and sentences. For example, in English, a question usually ends with a rising pitch, and the word "read" is pronounced very differently depending on its tense. Clearly, understanding how a word or phrase is being used is a critical aspect of interpreting text into sound. To further complicate matters, the rules are slightly different for each language. So, the front end must do some very sophisticated analysis.

The back end has quite a different task. It takes the analysis done by the front end and, through some non-trivial analysis of its own, generates the appropriate sounds for the input text. Older synthesizers (and today's synthesizers with the smallest footprints) generate the individual sounds algorithmically, resulting in a very robotic sound. Modern synthesizers, use a database of sound segments built from hours and hours of recorded speech. The effectiveness of the back end depends on how good it is at selecting the appropriate sound segments for any given input and smoothly splicing them together.

The developed speech synthesis engine instance uses Microsoft's Speech Application Programming Interface (SAPI). SAPI is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications. In general the Speech API is a freely redistributable component which can be shipped with any Windows application that wishes to use speech technology. Many versions (although not all) of the speech recognition and synthesis engines are also freely redistributable. Furthermore, all versions of the API have been designed such that a software developer can write an application to perform speech recognition and synthesis by using a standard set of interfaces, accessible from a variety of programming

languages. In addition, it is possible for a 3rd-party company to produce their own Speech Recognition and Text-To-Speech engines or adapt existing engines to work with SAPI. In principle, as long as these engines conform to the defined interfaces they can be used instead of the Microsoft-supplied engines.

The speech synthesis engine that has be developed to cover the speech output modality needs of the proposed platform uses SAPI to access pre-installed windows synthetic voices, Microsoft's Speech Platform's installed voices as well as voices from third party companies such as Nuance[9]. Furthermore it is able to adapt to fit the needs of the individual users as well. The following points present the capabilities, configurability and adaptability of the developed speech synthesis module:

- The auditory output of the system is based on artificial voice synthesis where the volume can be adjusted based on both the needs and the preferences of the users as well as the distance between the robot and its user or the levels of the surrounding environment noise. The user is able to define a minimum and maximum volume threshold for the speech synthesis engine so that the volume of the synthesized speech output gets adjusted accordingly. The further the robot is form the user, the higher the volume of the speech synthesis output is. When the robots gets "out of range" (according to a predefined value), the speech output is disabled. This is also the case when the user doesn't want to be disturbed or for some reason the robot shouldn't be permitted to talk (e.g. someone is sleeping in the same room). Furthermore, the volume of the produces speech output is adjusted accordingly to the surrounding environment ambient noise.

- In addition, the rate of the synthesized speech can also be configured to achieve a comfortable speaking pace based on the user's preferences or hearing acuity.

- Moreover, the option to choose either a male or a female synthetic voice is provided.

[9] http://www.nuance.com/for-business/mobile-solutions/vocalizer-expressive/index.htm

- Furthermore, the a-priori adaptation of the pitch of the sound output of the tablet PC of the robot can be adjusted from inside the operating system or by switching the output of the speech engine to local wav files which can be accordingly adjusted and then played back by the system.

- In addition, the engine raises events for the SpeechStarted and SpeechCompleted operations that are utilized at a higher level to switch on and off the speech recognition engine as described in 5.8.2.

- Finally, the runtime adaptation of the pitch of the speech synthesis output modality has been implemented based on the NAudio and SoundTouchNet open source libraries. In order to be able to control the pitch of the synthetic voice, the speech synthesis output modality has to be configured to provide its output as a .wav audio file. This file can then be opened with the help of the NAudio library, edited with the help of the SoundTouchNet library and played back using the inherent support of the .NET framework. NAudio is an open source .NET audio and MIDI library, containing dozens of useful audio related classes intended to speed development of audio related utilities in .NET. It has been in development since 2002 and has grown to include a wide variety of features. In the scope of the current thesis, it has been used as a means to open and read the created speech synthesis audio files. SoundTouch is an open-source audio processing library for changing the Tempo, Pitch and Playback Rates of audio streams or audio filesas well as it supports estimating stable beats-per-minute rates for audio tracks. The library was originally written in C++ while the SoundTouchNet wrapper has exposed the desired functionality to the .NET framework. The SoundTouch library is intended for application developers writing sound processing tools that require tempo/pitch control functionality. As a result, the library was ideal to cover our needs. The library was recompiled for the purposes of the current work with its parameters being optimized to handle speech input more effectively. The output raw sound input from the NAudio library was given as input to the SoundTouchNet wrapper which invoked the underlying library to adjust the pitch of the synthetic voice accordingly. The output was then written back to

a .wav audio file with the help of the NAudio library and played back using native .NET playback controls.

5.3.3 Gesture Recognition modality

Gesture recognition is the process by which gestures made by the user are made known to the intelligence system. Gesture recognition plays a significant role in Human Robot Interaction since it adds a natural dimension to the interaction process. People inherently use their hands when talking to convey their thoughts, intentions and feelings. Providing robotic platforms with a way to understand this kind of body language, opens new dimensions for intelligent household robotics that can understand their user more accurately.

The gesture recognition modality has been integrated into the proposed framework. The speech recognition engine that has been developed to cover the gesture recognition modality needs of the proposed framework is able to understand a predefined set of gestures that are relatively easy to perform and be remembered by the end users of the platform. FORTH's gesture recognition module [140], [141] provided by the CVRL lab has been used to this end. This gesture recognition module is subdivided into three more submodules:

a) A submodule capable of tracking the upper body joints, the hand and fingers and the full 3D skeletal body of a standing and sitting person. When tracking the upper body joints, the module detects the 3D position for the shoulders, elbows, wrists, head and torso joints (See Figure 19 left and Figure 21 right).
b) A submodule capable of tracking the person's full body. The tracker of this module detects additionally the 3D position of hips and legs joints (See Figure 19 right and Figure 21 right).
c) Finally, a submodule for tracking the hands and fingers of the person. The tracker of this submodule needs the 3D position of the palm and finger joints. (See Figure 20 and Figure 21 left).

Figure 19: Left: 3D skeleton detection and tracking of the upper body required for a sitting human in order to perform gesture recognition. Right: Full 3d skeleton-based body detection and tracking.

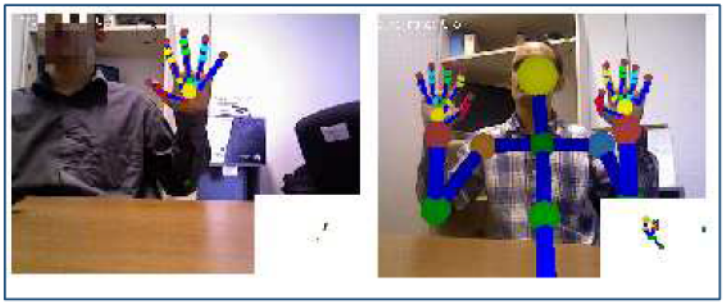


Figure 20 Left: Detection and tracking of only 1 hand and its fingers. Right: Integrated 3D detection and tracking of upper body and 2 hands/fingers.

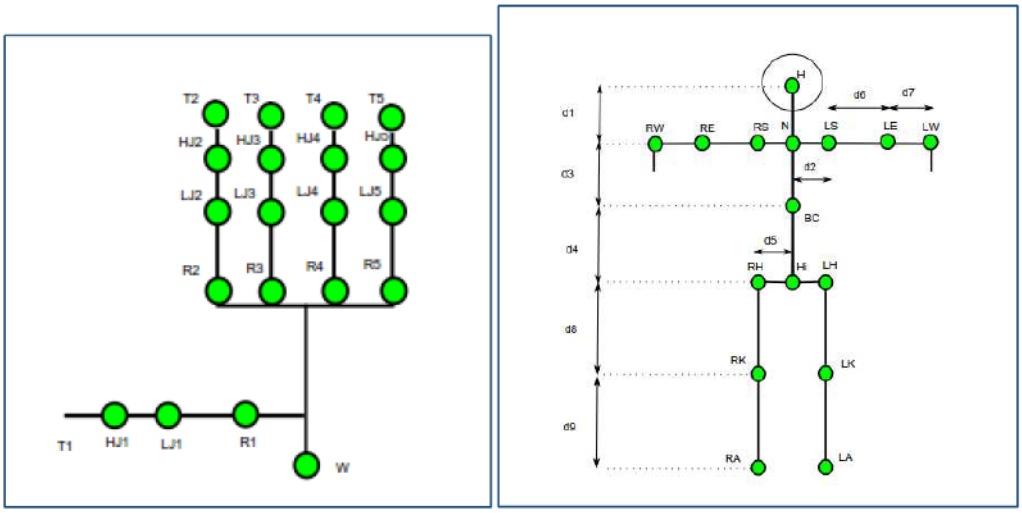


Figure 21: Left: Full skeletal-based hand and (right) body models used for 3D detection and tracking of body parts towards gestural recognition interface. Hand model involves 1 joint for the palm, 3 joints for the thumb and 4 per finger. The skeletal body model involves 3 joints per limb (for arms and legs) and 4 for the main torso (hip centre, torso, neck, head). An upper body model is a partial model involving the hip-center, torso and rest joints of the upper body.

Regarding the functional specifications of the gesture recognition module, gestures based on hand/fingers detection/tracking, have a working range of 0.6 – 2 meters with respect to the camera. For the needs of the tracker, only hands and fingers are

necessary to be observable by the camera. For gestures based on upper or full body detection/tracking their working range is 1 – 4 meters with respect to the camera. Finally, the camera should face the user's body front-o-parallel when being placed in a horizontal pose or can be placed higher and observe the user's body being tilted down up to 30 degrees.

The following table contains a sample subset of gestures that have been implemented [136] and their potential mapping to robot commands.

Id	User Command	Gesture/Posture	Robot command
1	**Yes**	Thumb up -palm closed (close-up- range interaction)	Provides a positive response to confirmation dialogues. YES gesture.
2	**No**	Index finger up and waving-palm closed (close-up- range interaction)	Provides a negative response to confirmation dialogues. NO gesture.
3	**Cancel**	Both open palms towards the robot (close-up-range or normal- range interaction)	Terminates an on-going robot behaviour/command.
4	**Pointing**	**The furthest arm from user torso pointing a direction in 3D space.**	Pointing to an object or place in 3D space using an extended arm for 2-3 seconds.
5	**Reward**	Circle gesture -open palm circular movement towards the robot (at least one complete circle is needed) (close-up- range or normal- range interaction)	Rewards the robot for an accomplished action/task.
6	**Emergency**	Cross hands pose (normal- range interaction)	Signifies an emergency situation.
7	**Come closer**	Raise either of the hands and open the palm toward the body. Bend the angle toward the body and back toward the robot 2-3 times.	Initiates a procedure for the robot to further approach a sitting user.

Table 2: Sample gestural vocabulary supported by the Gesture Recognition Engine

The gestures 1-2 require module C and optionally A. For module C, the 3D positions of the joints corresponding to palm and fingertips T1-T5 is required to be provided per frame in order for gestures 1-2 to be functional. Finally, gestures 3-7 require module A and optionally B.

In Figure 22 the set of the above hand and finger-based gestures is demonstrated.

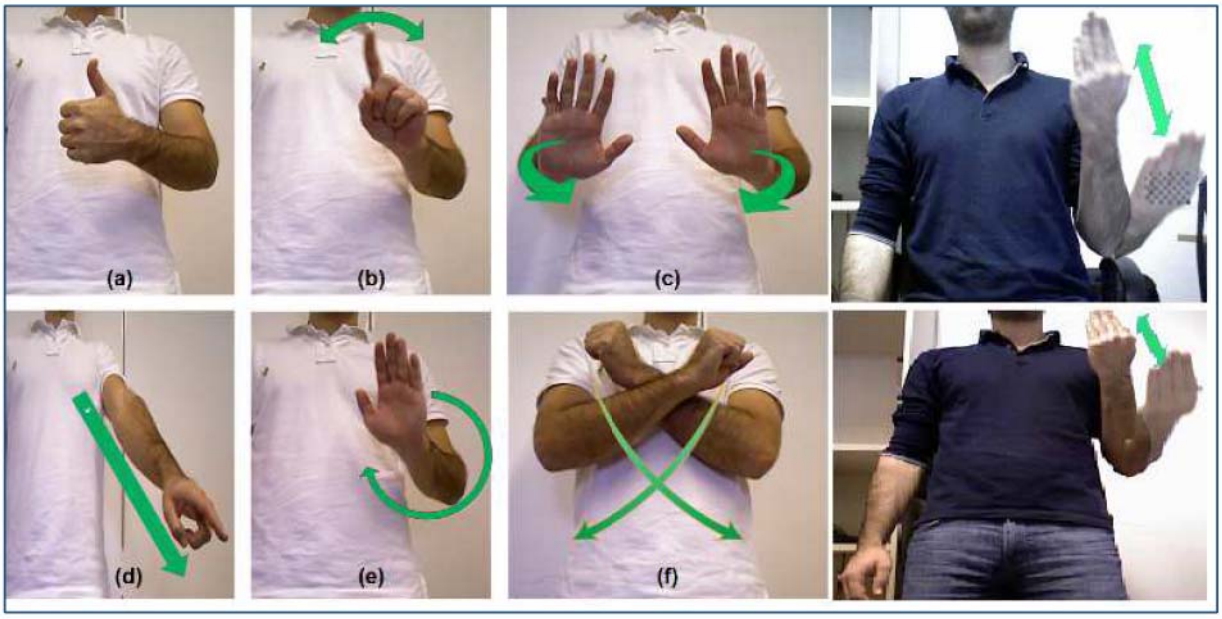


Figure 22: A sample of hand and finger-based gestures that are supported by the engine. (a) "YES" and (b) "NO" gestures utilized for confirmation dialogues. (c) "Cancel" command that can be used for close-up- and in-range interaction. (d) The pointing gesture indicates an object on the floor to be picked up. (e) A "reward" gesture can be performed to thank the robot for an accomplished task (one or more circles). (f) The "Help-me" gesture can be used to trigger an emergency task, (g) "Come closer" gesture, which issues the robot command to further approach a sitting user and enable reaching the touch screen if the robot was initially places too far away.

The gesture recognition module runs on Linux and exports the provided functionality through the ROS operating system. A bridge has been developed between ROS and Windows that run on the tablet PC in order to be able to integrate the gesture recognition results in the proposed framework. The bridge was implemented using ROS.NET[10] which is described in 5.11

5.3.4 Touch modality

The touch modality refers to the interaction that takes place between the human and the touchscreen tablet pc that is onboard household robotic platforms. Touch is an important aspect of human robot interaction because it consists a natural human approach. Selecting between desired items, reaching for different types of controls and adjusting various sensors are all part of humans' daily lives. The simulation of such daily activities can be done by using a touchscreen tablet PC that can be used both for output and input form the users to the robotic platform.

[10] https://github.com/nickkazepis/ROS.NET

The proposed framework integrates all the aforementioned modalities into a seamless set of interaction modes between the robot and its users. This results into a more natural nature of interaction, since the user is free to choose how to interact with the system based both on his/her preferences and the context of interaction. The robot can display its output on the onboard touchscreen device and use sound at the same time as redundant auditory feedback just like when people interact with each other. Furthermore, the robot is able to understand touches on the touchscreen device, gestures in front of the monitoring image acquisition sensors as well as speech commands given by the users. This provides redundant feedback provision which has been proved to be necessary especially when designing for the elderly user group [40].

Regarding the Graphical User Interfaces that are being produced based on the proposed framework, they are tailored to the needs of the end users.

The framework's building blocks have been designed based on the user-centric design principles and based on simplicity and clarity of the individual modes that each module represents (e.g. time selection module, binary decision module, multiple selection option module etc. [see 5.4]).

Furthermore, the used vocabulary can be easily adapted to the cognition levels of the users. The generated UIs are inherently translated into the user's native languages, with the translation files provided by the framework and fine-tuned by the users. In other words, the produced user interfaces are globalization and localization ready since the necessary language translation files are automatically generated by the system at runtime and can be edited offline by the developers of the developed applications.

Finally, the framework provides quick exit shortcuts to the main menu and access to emergency scenarios in an effort to increase the level of acceptance of this kind of technology and contribute to higher levels of user confidence and satisfaction while adding to the whole seamless user experience. This results into the user not being afraid or hesitating to use the system, since there are clear visible interaction paths that can take them to the desired functionalities of the system.

5.4 Multimodal UI framework

The slideshow approach

The software that runs on the robotic platform is a suite of applications and programs. Each application can be started from the main menu screen that acts as a placeholder for all available installed applications. Each application is a collection or sequence of different dialogue screens that are organized per application. Every dialogue can be handled in an autonomous way thus adding to the flexibility of the system while contributing to loose coupling of the architectural components.

The RobotApp Base Class

The proposed framework provides an easily extendable, modular, concrete set of UI components, base-classes, building blocks and tools for developing multimodal interactive applications targeted primarily to elderly users in the context of applications that run on a domestic robotic platform. In this scope, various adaptable and adaptive user interface components were designed and developed to support the creation of applications. The developed components fit together in a seamless manner that provide a universal look and feel for the created applications.

The framework was designed and developed having in mind that the set of applications that are going to be developed can be independent from each other. In other words, each developed application should be able to function on its own, register itself to the "application suite" of the robot, and manage its own state and status. All created applications should be provided as system plugins that can register themselves with the robot and become part of the system's add-on functionality. To this end, a base class, namely the *RobotApp* base class, was provided so that every newly created application should derive itself from it and use it as a stepping stone for building on it all the additional necessary functionality.

The applications' base class was designed to include support for frequent user actions and daily tasks that have multiple applications such as the need for setting up a desired time. Setting up a specific time is a task that is present in the everyday life of the users of our target group. The applications of a selected time can vary from setting up an alarm, scheduling a record command for a favorite TV program, setting up a timeout countdown to remember to check on a baking cake, of creating a reminder for the

intake of necessary medication. To this end the applications base class provides all the necessary functionality for specifying a desired time in the granularity of specifying the exact hour, minutes and am/pm segment of the selected time. This has been supported by the introduction of auxiliary helper classes that conform fully to the property change notification component model.

A very important piece of functionality that was implemented into the applications' base class is the capability of monitoring time and providing timed events. This can have multiple applications as the developers are given the possibility to include timestamps and timed events in their ACTA scripts. Applications that are composed of multiple UI dialogues can specify whether or not they need time tracking for each of their dialogues independently. Whenever the application transitions to a new state and a new UI component is displayed, the backend respect's the dialogue's preference for activating or deactivation the time keeping functionality and act accordingly. This process takes places transparently to the developer during the loading and unloading of the various UI dialogues. The time keeping functionality is implemented by the utilization of system timers that can become active or inactive according to the application's needs. The usage of system timers to support the tracking of time, introduced the need for conforming to the IDisposable system interface in order to provide the functionality for properly disposing and releasing managed allocated resources, namely the timers. Furthermore, the base class was required to conform to the property changed component model as well in order to be able to provide notifications that signal the invocation of the reasoning engine as discussed above in the previous sections.

Every created application is composed of numerous states and dialogues with each state usually corresponding to a different UI dialogue while a dialogue can match to one or more states independently. Applications are characterized by a starting state which is the initial state that the application is in when the system first starts up. Each created application is required to provide some basic information to the system, regarding its name, its icon and its starting state in order for the system to be able to recognize, initialize and display the application properly.

Furthermore, applications are provided with an instance of a Universal Translator engine that is used for automatically and transparently translating the various UI components into the user's native language (See 5.6) which is a process that automatically takes place during the UI initialization without the intervention of the programmer. In addition, a reference of the last displayed piece of UI dialogue that the application used to display on screen is kept in order to maximize performance without any significant memory overhead when applications come back to the foreground switching back from another system app, or when navigating back and forth from the main menu. When the application shuts down, the systems automatically creates any missing translations if instructed to do so by the application's configuration file.

Moreover, the base class provides applications with an instance of a Windows Workflow Foundation rule engine that can be used to run the rules that are generated from the application's ACTA script in order to implement adaptation scenarios or high level application functionalities. When the application is initialized, the WWF rule engine instance gets initialized accordingly and the application rules are loaded into the reasoning component. The application is then capable of loading additional rules on demand to fine tune or tweak its functionality.

Backing fields for the integration of the different modalities have been added to the applications' base class as well. The necessary arrangements have been made to accommodate for the implementation of the touch modality, the gesture recognition modality and the speech recognition modality in a way that it is extensible so that it can be modified to incorporate additional modalities in the future. Furthermore, a set of abstract base function have been provided which are already hooked to the different interaction modality events such as the recognition of speech or gestures. If the developer wants to incorporate a modality into his/her application, all he/she has to do is implement the corresponding modality callback function and populate it with the desired functionality.

The state in which the application is in at any given time is closely monitored and stored in the base class as well. Apart from the current state, the previous state of the application is also tracked as well as an additional special state holder, namely the

ReturnState register. This field is responsible for holding the desired state that the program should transition to, after completing intermediate tasks such as a time selection task or the display of an urgent notification on the screen. During the applications' state changes, the backend asks the permission of the Communication Planner (as discussed in 5.8.2) and then all the state fields are updated accordingly. The activation or the deactivation of the time keeping functionality is adjusted and the reason behind the state change is recorded if available. This contributes to being able to handle the state changes differently according to the reason behind them that caused the respective transitions. For example, it is different for the user to ask the robot to show him/her the time or ask the robot the question "What time is it?" in which case the robot should be smart enough to respond verbally as well, adding to the whole natural experience of the interaction.

Calling an application to appear on screen is as easy as displaying the last displayed dialogue of the application which is always saved in the base class for all applications, or if the application hasn't been initialized yet, instructing the application to transit to its starting thread which will cause the starting UI dialogue to be displayed because of the state transition.

Finally, the base class provides all the necessary virtual function for displaying application dialogues on the screen. These functions already contain some of the provided functionality for communicating with the Communication Planner for permission, or implementing frequent tasks' logic and functionality such as the time selection task which can be found throughout the daily life of the elderly.

The following sections discuss the different UI components that were developed.

5.4.1 UI components design and implementation

The RobotAppScreenBase Dialogue Class

While all apps that are built using the proposed framework must derive from the base class RobotApp as discussed above, all the dialogues that the applications create must derive from the RobotAppScreenBase base class. The RobotAppScreenBase class in turn derives from the UserControl class of the WPF framework, making all the created dialogues to be user controls that can be added to the main window and displayed on

the screen when needed. This dialogue base class provides all the necessary functionality to transparently support the automatic translations of the created dialogues, the integration of the different modalities as well as the automatic augmentation of all the used textboxes so that a virtual on screen keyboard automatically appears when the users tries to give input.

The RobotAppScreenBase creates a reference to the parent application to which each dialogue belongs in order to provide access to all the application's class members, functionalities and translations. When the dialogue is instantiated and loaded on the screen, it registers itself with the adaptation manager so that it can be automatically and transparently adapted to the preferences of the user, tweaked to match his literacy or experience regarding electronic devices, as well as adapted to the environmental context that the robot is in. Subsequently, the different modality hooks are created, namely the speech and gesture recognition modality hooks, so that the dialogue can recognize and respond to input coming from the various supported modalities. In the meantime, any potential speech grammar that are supported by the dialogue are loaded into the speech recognition engine and the parent application is informed whether the dialogue needs the time keeping functionalities that the application supports. Similarly to the above, when a dialogue get unloaded because either the application has something else to display, or it gets taken off screen in order to show the main menu, the opposite workflow takes place, unloading loaded grammars, unhooking modality hooks and unregistering the dialogue instance from the adaptation manager. Specifically regarding the different supported interaction modalities, the framework supports wired-up virtual functions (defined in the dialogues base class) which can be overridden in the dialogues in order to provide input from the modality engines.

The translation of the created dialogues is a process which is automatic and transparent to the developer. When a dialogue loads on the screen it is automatically translated to the current UI language by the provided translation engine of the parent application. To this end, there were provided all the needed auxiliary helper functions that support the traversal of the UI's visual tree, in order to find all the UI elements that can and were meant to be translated. These functions scan the entire visual tree of the dialogue, find all dependency objects, search for all translatable elements and

pass them to the translation engine. Text blocks and labels are inherently translated while the content of content controls is in turn traversed in search for translatable items as well. Finally, all components that conform to the created translatable code interface are instructed to initiate their translation process.

Finally, the dialogues base class contains the functionality for automatically augmenting the use of textboxes by hooking on their GotFocus event and dispatching a virtual on screen keyboard for providing the users with the means of giving input to the system. When the user touches a textbox, it get system focus end emits the GotFocus event. This event is captured by the base class and a new dialogue is automatically created, containing the textbox, the virtual keyboard and two buttons for accepting the changes or canceling the process. When either the changes are accepted or the process is canceled, the user is navigated automatically to the previous dialogue which contained the textbox, whose contents are now containing the input text. The process can be repeated as necessary by touching the textbox again. The auto-generated keyboard dialogue

The Virtual Keyboard

For the input needs of the target user group, there was a need for an on-screen virtual keyboard which could be translated to the user's native language while being adaptable and adaptive both to the users' needs and the context of interaction. Solutions like the WpfKeyboard[11] could not cover neither the adaptation needs nor the translation functionality that was needed. Their approach however modular in the sense of extensibility and scalability was too monolithic in terms of how it was displayed on the screen. Furthermore, that approach was not based on a virtual keyboard but rather on the utilization of a custom made input simulation engine[12] which was propagating the keyboard events to the operating system. However, since the windows operating system does not provide robust APIs to support such functionality, the discussed approach was based on, if anything, clever hacks which caused unexpected results and side effects when used with the latest operating

[11] https://wpfkb.codeplex.com/
[12] https://inputsimulator.codeplex.com/

systems including windows 8, 8.1 and 10. As a result a fully configurable, adaptive, adaptable and translatable keyboard was built from scratch.

Figure 23: The developed adaptable and adaptive custom virtual keyboard

The keyboard was built dynamically using CaseSensitiveKeys and TogglingModifierKeys which were both derived from a uniform LogicalVirtualKeyBase base class. The translation of the keyboard is merely a matter of remapping the content of the case sensitive keys to the content of the target language. The toggling modifier keys operate the same independently of the selected language. The keyboard user control conforms to the ITranslatable code interface so that it can automatically translate itself to reflect the selected language. The main keyboard is divede into four zones, namely four rows, each of which can contain multiple case sensitive or toggling modifier keys. The keyboard was built based on the touch input guidelines concerning older adults and touch devices. Springs were used as needed in order to provide the extra spacing between different parts of the keyboard. These springs are special buttons whose visibility becomes hidden at runtime. As a result, the keyboard takes its final form with the spring buttons being invisible to the end user while continuing to occupy the dedicated space since their visibility attribute is set to hidden instead of collapsed. The different button events are wired up in the base class of the keyboard while the display of the buttons is achieved using a uniform WPF grid of four rows one for each row of buttons. Button distribution is realized by special grid distribution attached properties so that each button can occupy a different amount of space on the screen which can be configured as needed. The final appearance of the buttons is determined by applied data templates and styles for the different involved user

controls, which is a very flexible and valuable functionality that is offered by the WPF foundation.

The Keyboard Input Dialogue

The created virtual on screen keyboard user control that is discussed above, was used to create a bigger application dialogue that was used for the augmentation of the framework's textboxes.

Figure 24: The virtual keyboard application dialogue used for the augmentation of textboxes.

Whenever the user clicks on any of the used textboxes on any dialogue that has derived from the RobotAppScreenBase dialogue class, the above dialogue is displayed on the screen to facilitate user input. The dialogue consists of a placeholder textbox (which is intentionally not augmented as this would cause an infinite interaction loop), an instance of the virtual keyboard and a couple of buttons that can be used for accepting or rejecting any changes made. The user can use the displayed virtual keyboard to input the desired text. Visual feedback is provided in the textbox above the keyboard. Upon accepting or canceling any changes, the user is taken back to whichever dialogue was displaying before the keyboard dialogue and any text is transferred to the respective textbox of that dialogue if the user had chosen to acknowledge the changes. Similarly to the keyboard user control, the special spring buttons that are used for real estate layout are hidden to the final users as the keyboard adapts to the context of interaction which is discussed in the adaptation section of this chapter.

Options Presenter

An options presentation user control was created to facilitate the case where the user needs to choose one among different options. The provided dialogue screen contains a panel that can be automatically populated with any amount of options. A representation of the options presenter dialogue can be seen in Figure 25.

Figure 25: The options presenter dialogue screen that can be used when the user has to select one among any amount of options.

The options screen is a dynamic auto-translatable application dialogue that is derived from the RobotappScreenBase base class. The different options are populated automatically through a developer provided collection of dynamic objects, each of which must provide a name, an icon, and a function that it wants to be executed upon its selection. It supports activation through the different available modalities and as such it accept a user-defined grammar to be loaded upon its display, to be used for speech or gesture recognition purposes.

The options pane can contain any amount of different items. The system automatically calculates how many of the available options can be displayed on the screen, how many pages of options have to be created and whether navigation buttons (i.e. forward and backward navigation buttons) have to be enabled or not. Predefined grammars are provided and can be automatically loaded by the system to support the backward and forward navigation. These grammars can of course be overridden by user-provided ones. The navigation buttons are automatically shown as active or disabled according to whether the user can navigate forward or backwards. This is

achieved by binding the UI buttons to backing dependency properties that get updated accordingly while automatically triggering the binding to update to reflect the changes.

Finally, the options screen automatically translates its caption and its back and forward buttons, however the contained items should be provided already translated because the content is context-specific and there isn't any apparent use case where it should be translated by the dialogue and not by the application that is using the dialogue.

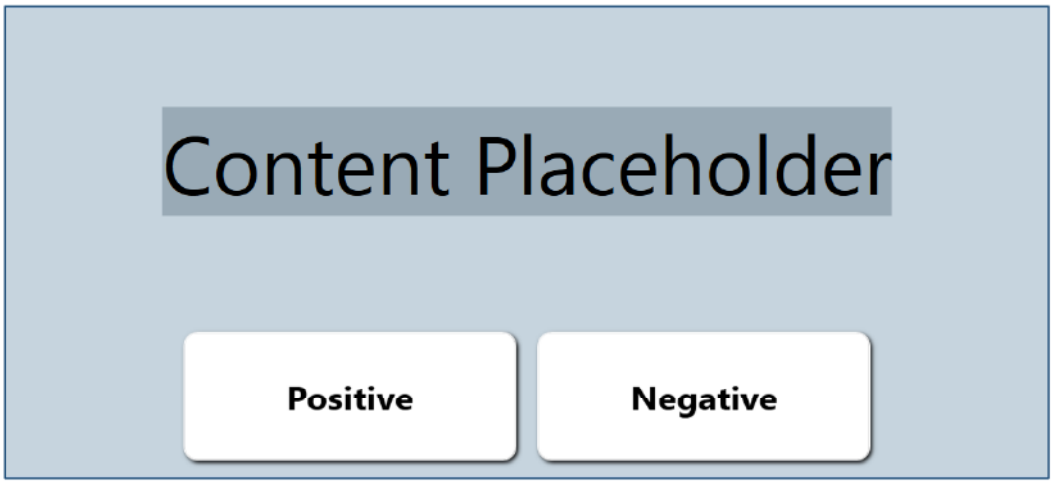


Figure 26: The binary decision framework dialogue containing the content over which the decision has to be made, the positive action button and the negative action button.

Binary Decision Dialogue

The binary decision framework dialogue is utilized in cases where the user has to make a decision between two things. This can refer to deciding whether to accept or reject changes, whether to add or cancel the creation of an alarm, as confirmation dialogues for destructive actions such as the deletion of data, etc.

The binary decision dialogue consists of a simple user control screen containing the content over which the decision has to be made and two buttons, one for taking positive action and one for taking negative action. For example, when adding a reminder, the content of the dialogue would be the user interface displaying the reminder itself, the positive action would be the saving of the reminder and the negative action would be canceling the creation of the reminder.

```
<DataTemplate x:Key="stringTemplate">
    <TextBlock Text="{Binding}"/>
</DataTemplate>
<fw:TypeTemplateSelector x:Key="TypeTemplateSelector"
                StringTemplate="{StaticResource stringTemplate}" />
```

```
<ContentControl Grid.Row="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
                VerticalAlignment="Center" x:Name="BDContent" Margin="50"
                ContentTemplateSelector="{StaticResource TypeTemplateSelector}"/>
```

```
public class TypeTemplateSelector : DataTemplateSelector
{
    public DataTemplate StringTemplate { get; set; }

    public override System.Windows.DataTemplate SelectTemplate(object item, DependencyObject container)
    {
        if (item is string)
            return StringTemplate;
        return base.SelectTemplate(item, container);
    }
}
```

Figure 27: The data template used to style string contents, the type template selector object instantiation (top), the content control object that is used as a placeholder for the dialogue's content (middle) and a fragment of the backend implementation of the type selector classes (bottom).

Figure 26 displays a generic binary decision dialog with the two available answers as well as the placeholder where the content (on which the decision has to be made) will be placed. The content of this dialogue can range from a single string or textblock to a however complex user control. To achieve this, a content presenter placeholder has been used (provided by the WPF framework) and the necessary type template selection helper classes have been developed to style the contained content accordingly. For example Figure 27 shows the data template that was used for styling strings, the instantiation of the type selector object, the content control that was used as a placeholder for the binary decision's content and a fragment of the code behind the binary decision that implements the type selector functionality for string contents. Finally, the binary decision dialogue is accompanied by a text to speech prompt which can be used for using the speech synthesis output modality and customizing the actual words that the programmer wants to be spoken to the user when the dialogue is displayed.

Notification

The notification dialogue is a dialogue capable of displaying any content in terms of notification. It derives from the RobotAppScreenBase base dialogue class as well and it provides the functionality for being automatically dismissed after a predefined configurable amount of time. To achieve this, the notification dialogue utilizes the time keeping functionalities of the base class. The time parameter that specifies the

timespan for which the dialogue will be visible on the screen can be easily specified in the ACTA script of the application. Similarly to the binary decision dialogue it supports the text to speech output modality that can be configured if the developer of the application wants the robot to be able to read the notification text to the user.

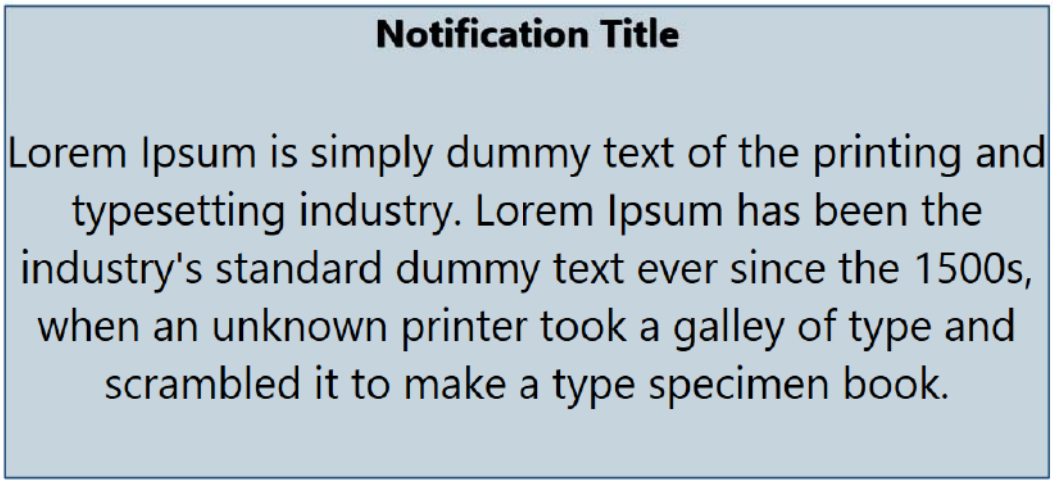


Figure 28: A sample auto-dismissable notification framework dialogue.

Spinner

The spinner user control was developed to facilitate the user when he has to choose among numeric or literal values. It comprises a caption, a selected value and two buttons, one for moving on to the next value and one for moving on to the previous one. It can work both for numeric and string arguments.

When it is used for numeric values, a minimum and a maximum value can be specified as well as the behavior of the control when it reaches one of the two ends of the value field. The developer can specify whether the behavior will be that the control will wrap around the edges or simply stay to the corresponding limit. Dependency properties can provide the minimum, the maximum as well as the currently selected value of the control at any given time. The selected numeric value in particular, is also provided in a string representation format for convenience purposes when it is to be used with data bindings in the UI frontend.

On the other hand, when the spinner is to be used with literal values, an ordered list of the possible string values must be provided to the control. The developer is then

allowed to choose one of these values as the initial value, while the two buttons when pressed cause the control to move on and display the previous or next value in the value list.

Figure 29: The spinner user control. Numeric values in black, literal values in orange for readability purposes.

The spinner user control does not derive from the RobotAppScreenBase base class as it does not need the functionality that the class provides. On the contrary, it is derived directly from the WPF foundation's UserControl class in order to be able to be embedded in bigger user controls or robot application dialogues. The control registers itself with the adaptation manager on initialization so that it can adapt to the colors and sizes of the application based on the context of the interaction. Furthermore it complies with the ITranslatable code interface and the INotifyPropertyChanged interface component model. The former ensures that the control will be able to translate itself based on the translation engine provided by the parent application it belongs to, while the latter provides the necessary notifications when the value changes. A developer can specify the mode of usage (numeric or litteral), provide the necessary values, hookup a callback to be triggered when the value changes so that the change can be handled properly and then include the control inside either an ITranslatable bigger user control, or directly inside a dialogue that derives from the RobotAppScreenBase class. This ensures that the translation of the control will take place transparently and automatically.

Figure 29 depicts a sample spinner user control. The numeric value is displayed in black and the literal value is superimposed in orange. During runtime only one of the two is visible and adapted to the color scheme of the application. The other one's visibility mode is set to collapsed based on the selected mode of operation.

Calendar

The calendar user control is a user control that displays a calendar showing the selected month, year and day of the month. Similarly to the spinner control it is not derived from the RobotappScreenBase because it does not need the functionality that it provides. It is however an ITranslatable and INotifyPropertyChanged compliant user control that can be embedded into other controls and dialogues.

Figure 30: The calendar user control with three modes of interaction for selecting either the desired day (left), the month (middle) or the year (right).

Figure 30 shows the three modes of operation for the calendar user control based on the desired mode of operation. The first is for selecting the desired day, the second is for selecting the desired month and the third (on the right) is for selecting the desired year. All three modes provide the necessary notifications in compliance with the notification changed component model. The whole control has been embedded into a viewbox so that the size of the control can be adjusted at runtime to fit the adaptation parameters of the application. Finally the color scheme of the control can be adjusted at runtime as well based on the adaptation properties of the application at any given time.

Tile Button

The tile button user control is a user control representing a button that has been shaped into the form of a tile. It contains a background image and a label that describes the context of the button. It can be activated by touch speech or gesture by developer defined vocal commands or gestures in the applications' ACTA scripts.

Figure 31: The tile button comprising a background image and a label. Tile buttons can have different appearances, colors and sizes based on the respectively selected styles and can be activated using any of the available modalities as specified into the application's ACTA script.

Figure 31 depicts a sample tile button showing a background picture of lightbulbs and displaying the label "lights". The tilebutton's control template has been defined in the active style that is being used by the application so that it is easy to be overridden by a new style provided by the developer for selected applications or for different tasks of the same application. Upon press, the button informs the parent application to whom it belongs that it has been pressed so that the application can process the event at a higher level through ACTA scripting, taking in to account any other modalities or context parameters. For example, if the "switch on" button has been pressed while the room lights are already on, the application should ignore it, or if the button can be pressed only when the speech modality is disabled, the application can take action accordingly based on the input.

Regular Buttons, TextBlocks and TextBoxes

The framework does not provide special implementations for regular buttons, text blocks or text boxes. The developer is free to use the implementation provided by the Windows Presentation Foundation based on the framework's guidelines. These guidelines dictate that if a textbox is placed inside a RobotAppScreenBase derived UI dialogue that it will be automatically augmented to display the on-screen virtual keyboard when it gains the focus of the dialogue. Similarly, any text block that wants to be automatically translated by the framework should specify either a Name or a

UId from Microsoft's winfx XAML scheme[13]. The automatic translation process is described in 5.6.

Time Selection

The time selection task has been inherently supported by the framework as it is a necessary functionality that is involved into many daily tasks in the everyday lives of the elderly. Tasks ranging from setting up a reminder for medication intake purposes, to setting up a cooking countdown timer or simply setting up an automated recording of one's favorite TV show, they all require the user to be able to define a desired timeframe. Based on the cognitive condition of the user and his previous experience with digital devises, the time selection task can be a hindrance that will discourage users and negatively impact their acceptance towards new technologies. In order to make the developed applications easier to use, more intuitive and easy to understand and as a result the robotic platforms more convenient to use, the time selection functionality has been integrated into the framework.

The framework provides an easy and intuitive way to select a specific time to be used with the developed applications. The time selection process is a dynamic adaptable and adaptive workflow that can be triggered by the developer simply by calling a function, namely the *InitiateTimeSelection()* function, from inside the corresponding application's ACTA script. A set of user controls have been designed and developed to facilitate the process of selecting the user's desired time. These controls are orchestrated into a series of dialogues that guide the user through the selection process.

The selection process is optimized according to the user's experience and preferences, contributing to a customized solution that fits the user's needs. Having a uniform workflow to facilitate the time selection process, contributes to a smoother learning curve because the users of the different applications will face the same familiar process when it comes to selecting a desired time. This in turn leads to the increased ease of use, ease of learning, subjective satisfaction and overall acceptance of using new technologies.

[13] http://schemas.microsoft.com/winfx/2006/xaml

The framework contains different controls that facilitate the time selection process. These controls are adapted at runtime on demand to cover the needs of the corresponding application that is using the functionality as well as the context in which the functionality was needed. The developed controls cover the different potential levels of user experience. The selection of controls that will be displayed is chosen at runtime based on the adaptation parameters. The user is guided through the time selection process while the option for navigation between the different steps (selection of hour, selection of minutes, etc.) is given when the process is divided into multiple dialogues.

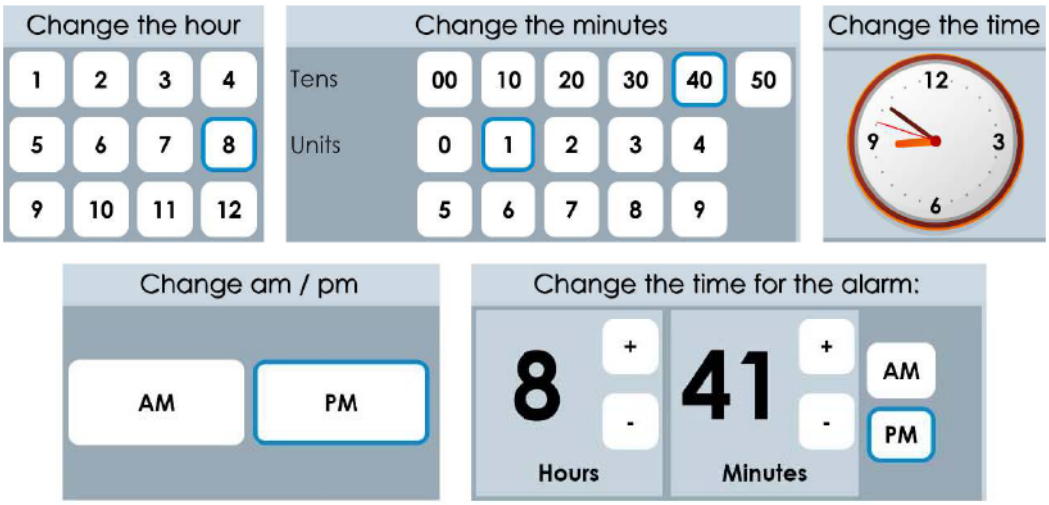


Figure 32: The various controls that support the process of time selection. Hour, minute and am/pm selection (top) and intermediate to expert options (bottom).

The developed controls cover the different potential levels of user experience. The selection of controls that will be displayed is chosen at runtime based on the adaptation parameters. The user is guided through the time selection process while the option for navigation between the different steps (selection of hour, selection of minutes, etc.) is given when the process is divided into multiple dialogues.

The novice hour, minute and am/pm selection controls as well as two other options for the intermediate and expert user can be seen in Figure 32. Novice options independently for hours, minutes and am/pm is also provided, using spinners that operate on literal values, in order to facilitate different user cognition levels. The time selection controls are simple user controls that do not derive from the

RobotAppScreenBase class. They are registered with the adaptation manager during their initialization time, and they comply with the ITranslatable model in order to be translated during runtime. Similarly to spinners, they implement the notification changed component model mechanism so that they can provide notifications when their values change during the interaction. Their values and captions are offered through dependency properties so that they can be used in data bindings in the frontend. Finally, they can be used by developers outside of the time selection context as needed, since they are components of the framework.

5.4.2 UI Navigation and scene management

The different UI controls and framework elements must be embedded inside a window in order to be displayed on the screen. This is a process that has been automated by the framework. The proposed framework provides the main window for displaying applications and hence their UI dialogues and elements.

The main idea behind UI navigation is that there is only one window capable of displaying information. This window is being managed by the Scene Manager who is responsible for updating the content that is displayed at any given time. The main window must comply with the INavigator code interface for the Scene Manager to be able to update its content. The INavigator interface contains all the necessary functions for updating the content parts of a Navigator window. As a results, applications can't display themselves directly on the screen, but have to ask the Scene Manager to do so. When an application wants to display a UI dialogue, it asks the Scene Manager to display it on the main window. The Scene Manager in turn, asks for the permission of the Communication Planner as discussed in 5.8.2 and acts accordingly.

This section discusses the main window that displays the different elements, the main menu application as a means for housing all the available applications and the management of the different application dialogues or user scenes.

UI Navigator

The main (root) window that displays all the framework UI elements, components, dialogues and applications is the UI Navigator window. It provides a bottom ribbon for

constantly displaying a shortcut to the main menu and controls for activating emergency scenarios. It also displays visual cues that can inform the user whether the different interaction modalities are available and what is their state. For example, when the robot is providing speech output to the user a corresponding visual cue is activated on the screen to provide visual redundant information that the robot is speaking. Furthermore, while the robot is speaking the speech recognition modality remains disabled, which is also show in the screen. The above apply to all available modalities and can be easily extended to support extra modalities in the future.

The main part of the UI Navigator is a content placeholder that can present content. For the needs of content presentation, the INavigator code interface was developed that contains the necessary functions for displaying content inside a UI Navigator window. In this scope, the developer is free to either choose the navigator window that is provided by the framework, or create a separate root window which complies with the INavigator interface.

When a Navigator Window is instantiated, it must register itself with the Scene Manager so that the manager knows where to show the requested content. Furthermore, the scene manager is able to display content by using the functionality described in the INavigator interface that every navigator window implements.

The main menu is an application just like all others that has a special purpose. Its purpose is to house all available applications and display them in terms of a menu. Furthermore the main menu is the application that is visible on the display when the systems starts. As a result, the navigator window must be aware of the existence of the main menu in order to be able to display it. However, this applies only for the main menu, since all other applications can be automatically discovered and loaded by the main menu itself.

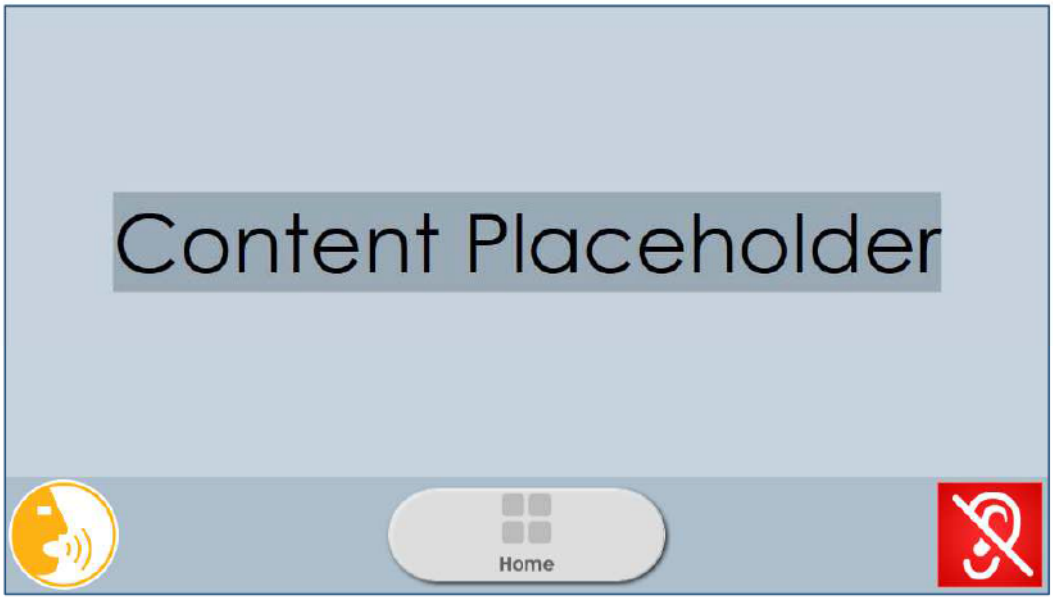


Figure 33: The framework provided Navigator window showing the content placeholder area, the ribbon area and sample home button and modality status visual cues.

The navigator window that is provided by the framework implements the INavigator, ITranslatable and IDisposable interfaces. The first is for being able to be manipulated by the Scene manager, the second is for being able to be translated (the main menu's translation engine is used for the translation of the main window) and the third is for being able to properly disposed managed resources, namely the instance of the main menu it utilizes (since the main menu instance is an application, it derives from the RobotApp base class which in turn contains the timers which are responsible for the time keeping functionality of the framework and hence have to be disposed-off properly).

The navigator window is responsible for initializing the scene manager and registering itself with it, as well as initializing the Communication Decision Maker, the Adaptation Manager, the available modalities and the Main Menu application. Furthermore it registers itself with the adaptation manager, loads the main menu and initializes the visual cues that show the states of the available modalities. These states are provided as dependency properties to facilitate data binding in the frontend. Finally, the activation and deactivation of the different shortcut buttons inside the ribbon is realized by the WPF routed commands input mechanism.

Main Menu

The Main Menu is a special application created for the proposed framework, whose purpose is to discover, host and display all the available applications. The discovery of the applications is based on reflection and the display is realized through the use of the Options Presenter framework dialogue. The main menu application leverages the navigation among the different menu pages from the options presenter in order to have better overview and control over what is being displayed at any given time.

The main menu application is just like any other application created based on the proposed framework. It derived from the Robotapp base class and uses all its functionality regarding automatic translation, ACTA reasoning, state traversing as discussed above. Upon initialized, it scans the root directory where all the available applications exist, and checks for any library applications that derive from the RobotApp base class except for the main menu itself. All discovered applications are hosted inside an internal list and displayed on the screen with the use of tile buttons as an instantiation of the options presenter dialogue. The concurrently displaying number of applications is configurable, the navigation between the menu pages supports all the available modalities, and the dialogues are displayed through the ACTA scripting mechanism.

Scene Manager

The scene manager is the component responsible for displaying content on the root window of the framework. It contains a reference to the register UI Navigator window and it is aware of the INavigator interface functionality that is implemented by the navigator window. The display of content is completely asynchronous to prevent race conditions between the different applications that may be trying to display their content simultaneously. Finally, it is based on a lazy instantiation, double locking version of the singleton software design pattern to ensure that only one instance of the class will be active at any given time and that it will be shared among the different available applications.

5.4.3 Abstract UI representations and UI generator

The INNTEACT framework provides a set of abstract user controls to facilitate the process of time selection as discussed above. These abstract controls comprise the

high level functionality for selecting the hour, the minutes and the am/pm part of a desired time which can then be used in whatever context is needed by the respective application. The selection of hierarchies that were realized includes workflows for the novice, intermediate and experienced user where the selection of actual concrete user controls take place at runtime according to the ACTA script that has been developed in order to support this functionality. The time selection ACTA script contains all the necessary information that is required by the system to decide which controls are going to be instantiated, how they will be grouped together, which application dialogues will be shown and how the whole time selection process will be orchestrated.

The logic behind the time selection task has been embedded into the framework. The reasoning engine that is needed to run the ACTA script in order to make the necessary decisions is the respective application's windows workflow foundation rule engine instance. The time selection ACTA script is loaded into the engine on demand, and unloaded when the process has completed. Based on the ACTA script, the application is able to decide at runtime on the task hierarchy that is going to be used and then instantiate the different corresponding user controls, group them into dialogues and provide the necessary navigation from one dialogue to the next. Finally, the whole process is completely transparent to the developer who just needs to call the initiation function to trigger the activation of this process.

The above functionality is inherently supported by the proposed framework and can be used to implement different adaptive component hierarchies that can be embedded into the framework as well.

5.5 Adaptation

Following the user-centered design approach the elderly user group has been given extensive attention at each stage of the design process. The framework's elements, components, library controls and high level dialogues can be optimized around users' needs and preferences rather than forcing the users to adapt their interaction patterns to accommodate arbitrary design decisions. Moreover, the proposed framework has been based on the concepts of design for all and universal access to

produce an adaptation mechanism that will be able to change the appearance of the generated user interfaces based on the profile of each user. This is of great importance, because the elderly users form a very heterogeneous group as each user might experience a different set of impairments that have to be taken into account. As a result, numerous combinations of functional limitations can be addressed such as motor impairments, vision impairments, hearing impairments, etc.

The proposed framework supports automatic user interface adaptation based on both off-line (user profile) and on-line (user interaction monitoring) knowledge. The generated interfaces are tailored to the end user based on both the adaptability (off-line adaptation) and adaptivity (on-line adaptation) concepts. A rules engine is used at runtime, to run rules against an ontology specification of the target user population that infers facts about the active user who is interacting with the current/respective application. These facts are being used by a decision making component to activate or deactivate the appropriate components of the adaptive component hierarchies on which the user interfaces are generated. The above methodology ensures that the end-user will experience an interaction experience tailored to his individual user attributes and to the particular context of use.

The functional and non-functional user requirements elicitation of the primary user group which is the elderly has been realized through the means of literature review as well as through extensive studying of the results of the HOBBIT and RAMCIP research project findings. Taking into account the research findings from the relevant literature along with the insights that have stemmed from the two aforementioned research projects has helped build a valuable framework in the hands of application developers.

5.5.1 Adaptation logic

One of the main reasons for the increasing importance of adaptation is that we interact with our applications in contexts of use which are more and more varied because of the advent of mobile technologies and smart environments. Various aspects can be part of the possible contexts of use and can be grouped along four dimensions (see Figure 34):

a) user-related aspects: preferences, goals and tasks, physical state (e.g. position), emotional state, etc.;
b) technology-related aspects: screen resolution, connectivity, browser, battery, etc.;
c) environment-related aspects: location, light, noise, etc.;
d) social aspects: privacy rules, collaboration, etc.

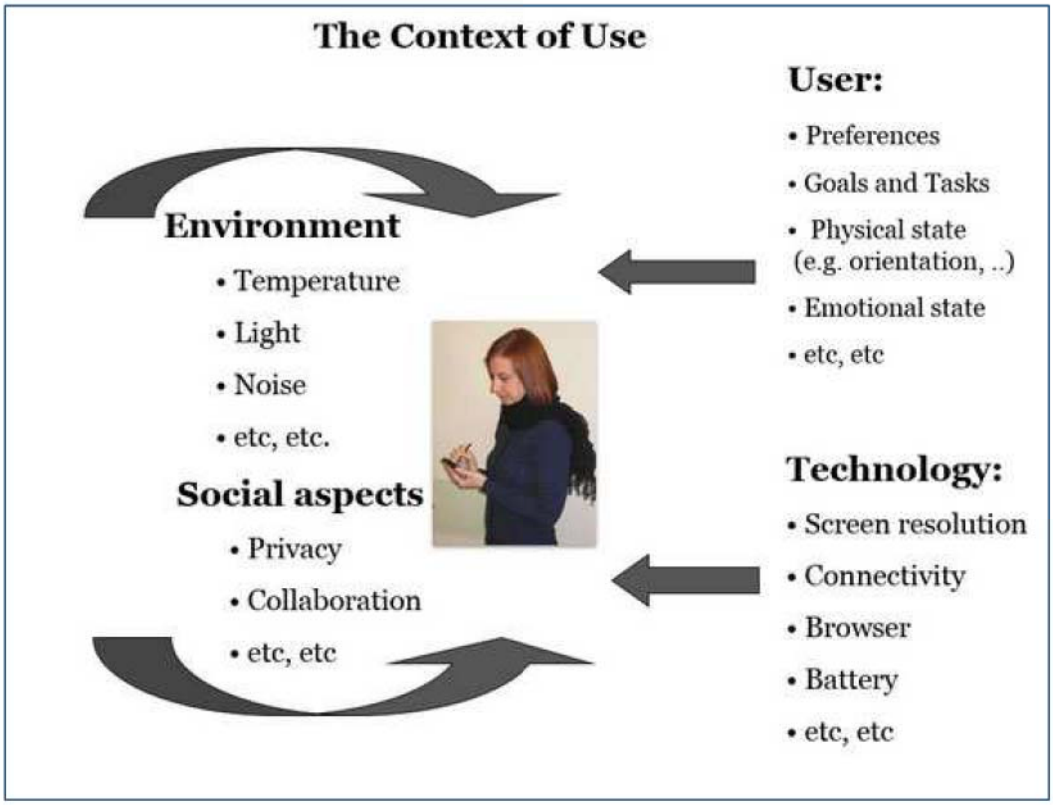


Figure 34: The context of use. Image: Courtesy of Fabio Paterno. Copyright: CC-Att-ND-3 (Creative Commons Attribution-NoDerivs 3.0 Unported).

According to changes in those aspects of the context of use any aspect characterizing a user interface can be modified. Thus, the user interface can be adapted in its: presentation—the perceivable aspects, including media and interaction techniques, layout, graphical attributes; dynamic behavior, including navigation structure, dynamic activation, and deactivation of interaction techniques; and content, including texts, labels, and images.

Various adaptation strategies are possible, which can be classified according to the impact they have on the user interface: conservation, e.g., simple scaling of UI elements; rearrangement, e.g., changing the layout; simplification / magnification, same UI elements but with modified presentation; increase (also called progressive enhancement) / reduction (also called graceful degradation), in terms of UI elements [142].

The proposed framework supports adaptation in terms of both adaptability and adaptivity [143]. The former ensures that the applications adapt to the preferences and limitations of the user so that he is presented with a full usable interface to interact, while the latter is based on monitoring the interaction history and adapting the applications to dynamically changing factors or inferences e.g. the ambient noise or the difficulties that the users are facing with the corresponding user interface instance.

The primary objective of this work has been to provide developers with a framework that facilitates them to build multimodal interactive applications that target the elderly user group and are to be deployed on household robotic platforms. In this scope, the different functional components of domestic robotic platforms have been taken into account both in terms of supporting them and taking them into account in terms of adaptation. The rest of this section presents some of the basic functional components of domestic assistive robots, their functional role that can be supported by the proposed framework as well as the adaptation aspects that concern them.

5.5.1.1 *Modality selection*

The different modalities that are supported by the framework can be activated or deactivated individually according to the preferences of the users and the context of interaction. The framework can decide on the optimal set of modalities to enable, fine-tune and fuse together in order to provide the end users with an interaction as seamless and as natural as possible. Furthermore, the selection of the different modalities and their fusing, is transparent to the developer as it is handled automatically by the framework. The developer has full control over which modalities are going to be supported at any given time as well as when and how they will be

activated or deactivated. However, the developer is also given the opportunity to provide the basic functionality that he wants to make available to each of the aforementioned modalities and then let the framework decide on how and when each modality gets activated. For example, the developer can explicitly specify which parts of his application can benefit from a specific modality and which parts must be contained only to specific modes of interaction. He can specify when he wants only a specific modality to be used or when any input from any of the available modalities can be considered valid. For example he can enforce that for critical application decisions, only the touch modality will be considered a valid way of confirmation, while for all other parts, any speech or gestural input will be allowed to be interpreted and treated accordingly. Finally, the framework is able to handle tricky cases where one modality might have to be deactivated due to dynamically changing conditions, although the developer has allowed its input. For example, the speech modality might have to be deactivated in noisy environments, or the gesture recognition modality might have to be deactivated in situations where the environment light is insufficient.

5.5.1.2 UI adaptation

The proposed framework supports adaptation through both adaptive component hierarchies and adaptive style hierarchies. The former is based on the design and implementation principles of unified user interfaces while the latter is based on the use of adaptive style hierarchies, as they are supported by the windows presentation framework, to either specify the desired application coloring scheme and sizing guide for the different controls and UI elements, or change completely the different framework elements' appearance.

5.5.1.2.1 Adaptive Component Hierarchies

Adaptive component hierarchies are inherently supported by the proposed framework. The supported tasks that have been implemented, have been described in an abstract manner at a higher level while general guidelines have been provided according to their instantiation strategies. For example the time selection procedure has been incorporated into the system and has been described at a higher level using ACTA. The time selection task has then been declared to comprise the consequent

selection of hours, minutes and time specifiers according to the time of the day. General guidelines have been given according to the expertise of the user that is using the respective application based on his profile. These guidelines specify how the whole task of time selection can be orchestrated in order to be presented to the user who is going to be guided through the process of time selection. Furthermore, user preferences have been taken into account, so that specific user control should be used or omitted during the process. Finally the whole task is realized in a transparent to the developer manner who can simply declare that he needs the time selection process at the desired place inside the applications that he builds. The initiation of the task takes place automatically, and the developer can explicitly declare the starting and ending state and consequently the starting and ending application dialogue that will be displayed to the end user.

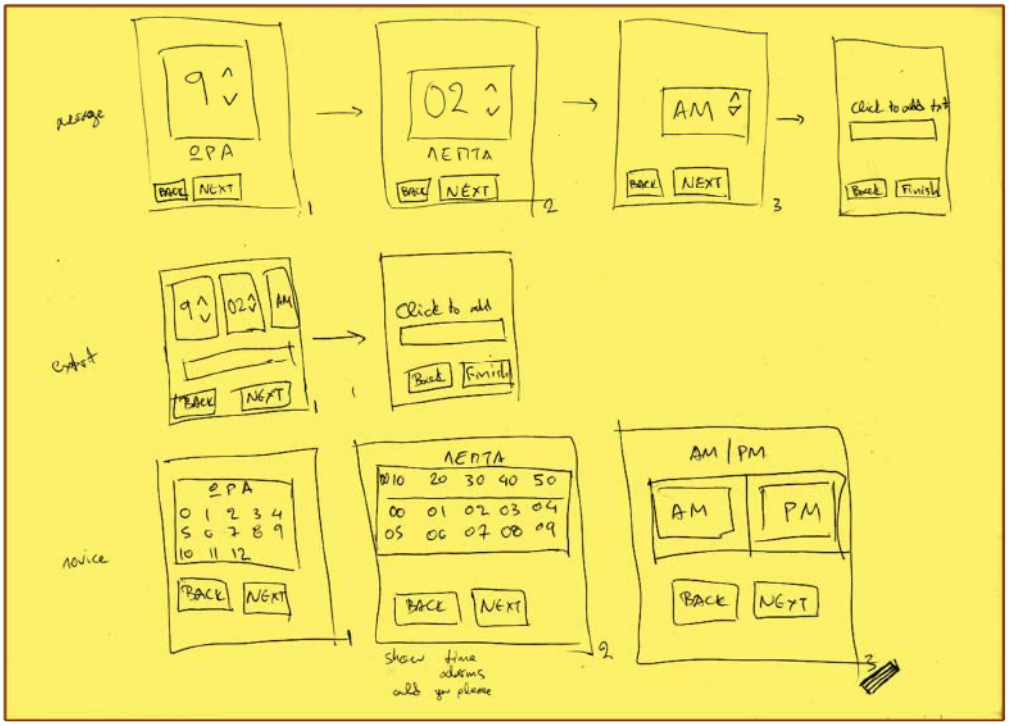


Figure 35: Time selection adaptive component hierarchy low fidelity paper based prototyping

Figure 35 depicts the time selection adaptive component hierarchy during its early low fidelity paper based design process. The different controls and UI dialogues that comprise the time selection component hierarchy have been selected and adapted

accordingly. The above design led to the final form of the time selection adaptive component hierarchy that has been incorporated into the FIRMA framework.

5.5.1.2.2 Adaptive Style Hierarchies

In addition to the adaptive component hierarchies' principles and design guidelines, the approach of adaptive style hierarchies has been adopted. According to this approach, the sizes and colors of the displayed framework elements can be controlled by styles that can be applied both at design time and at runtime. A number of cascading stylesheets have been developed to be used in the context of adaptation based on this approach. A subset of the developed styles have been used during the design time so that the developer can have a clear understanding of the appearance of the different user controls and dialogues that he/she is incorporating into the developed applications. The design time styles collection has been consolidated into a single higher level style file which can be included in the designed user controls and dialogues.

```
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
>
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="StylesBase.xaml" />
        <ResourceDictionary Source="DarkColors.xaml" />
        <ResourceDictionary Source="BigSizes.xaml" />
    </ResourceDictionary.MergedDictionaries>
</ResourceDictionary>


<fw:UCRobotAppScreenBase.Resources>
    <ResourceDictionary>
        <ResourceDictionary.MergedDictionaries>
            <ResourceDictionary
                            Source="Styles/DesignTimeStyles.xaml"/>
        </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
</fw:UCRobotAppScreenBase.Resources>
```

Figure 36: The design time styles file (top) and its incorporation during the development of a dialogue that is based in the RobotAppScreenBase class (bottom)

Figure 36 (top) shows the design time styles file as a resource dictionary that consists of multiple merged children resource dictionaries which define the appearance (based

on the specified visual tree of declared elements), the coloring scheme and the sizing dictionary that is going to be used during the design time. Furthermore, the incorporation of the design time styles file into the design of a new application dialogue that derives from the RobotAppScreenBase class can be seen in the same figure (bottom). The design styles file is embedded inside the resources of the designed dialogue, in order to enable the sample appearance that is defined inside the individual children resource dictionaries. This applies only during the design time. During the runtime, these resource dictionaries are being replaced by the runtime rules and adaptation strategies of the adaptation manager functional component which is able to load and unload the appropriate styles at runtime to suit the needs and preferences of the users as well as fit the context of interaction. The design styles are superseded by the runtime style, which in turn are dynamically loaded and unloaded during the interaction process.

The adaptive style hierarchies that are used for adaptation purposes during the runtime have been split into three major categories. The first contains all the styles that handle how the different framework elements will be displayed. These styles contain all the individual stylistic decisions that drive the appearance and define the visual tree of all the framework elements such as buttons, lists, dialogues, text entry controls, labels etc. The second category contains all the styles that define the coloring scheme of the application including foreground and background colors for all framework elements, border brushes of the different user controls, darker backgrounds for giving emphasis to specific UI elements etc. Finally the third major category contains all the styles that correspond to the sizing decisions of all the framework elements and UI dialogues, including button sizes, dialogue sizes, virtual keyboard sizes and margins, text input control sizes etc. The appearance of the final user interface is decided at runtime by the adaptation manager through a process of "pick and match" among the different available cascading adaptive style hierarchies, by selecting one from each major category. As a result, one style for visual appearance is selected, one style that defines the coloring scheme is placed on top of that and finally one more style that defines the overall sizes of every element is superimposed on the selection for filling in the missing sizing information and restoring the dynamic

bindings between all three style collections. As a result, every style can refer to any other category of styles through the use of dynamic resources declarations. This means that each style is only responsible for the category in which it exists while being allowed to contain bindings across different categories. Hence, the visual appearance styles can contain bindings to the sizing category styles which are going to be realized once the specific sizing resource dictionary that is going to be used, has been defined and linked to the runtime of the framework. This approach can create an arbitrary number of application appearances based on the selection of the activated styles and the possible combinations among them. For example, if there are three different styles defined in each of the three different major style categories, the developers can choose among any of the twenty seven combinations (27, i.e. 3x3x3) of the available UI instantiations. However, the selection of the developers are being superseded by the adaptation manager decisions as deemed necessary during the runtime.

5.5.2 Decision making

5.5.2.1 Windows Workflow Foundation

Windows Workflow Foundation (WF) is a Microsoft technology that provides an API, an in-process workflow engine, and a re-hostable designer to implement long-running processes as workflows within .NET applications. A workflow is a series of distinct programming steps or phases. Each step is modeled in WF as an Activity. The .NET Framework provides a library of activities (such as WriteLine, an activity that writes text to the console or other form of output). Custom activities can also be developed for additional functionality. Activities can be assembled visually into workflows using the Workflow Designer, a design surface that runs within Visual Studio. The designer can also be hosted in other applications.

With the availability of Windows Workflow Foundation (WF), Microsoft is introducing new rules capabilities to the Windows Framework Extension developer platform, the application programming interface (API) used in Vista and the forthcoming version of Windows. These capabilities extend from simple conditions that drive activity execution behavior all the way up to complex RuleSets executed by a full-featured forward chaining rules engine. Rules technology is exposed in two key ways in WF—

as conditions on activities and as a forward chaining RuleSet in the Policy activity. Forward chaining refers to the ability for the actions of one rule to cause other, dependent rules to be reevaluated. Forward chaining is a very powerful notion that allows atomic rules to be assembled into RuleSets without the definition of, or necessarily even the knowledge of, the dependencies among the rules. However, in some scenarios, the rule writer may want the ability to provide more control over the chaining behavior, specifically the ability to limit the chaining that takes place. This enables the rule modeler to limit the repetitive execution of rules, which may give incorrect results, increase performance and prevent runaway loops. This level of control is facilitated in WF rules by two properties, a Chaining Behavior property on the RuleSet and a Reevaluation Behavior property on each rule. The exploitation of these two properties in terms of manual control over which rules are going to be executed has made the WF rule engine component an invaluable tool for building applications that have the ability to reason and infer facts that can drive the application's logic. The ACTA scripting language runtime extension has been based on the manual activation of the rule engine and the manual management of the chaining behavior of the ruleset.

5.5.2.2 Adaptation Manager

The decision making component of the proposed framework in terms of adaptation, namely the adaptation manager, has been based on the rule engine provided by the windows workflow foundation framework. This rule engine provides all the necessary functionality for defining and declaring all the different adaptation decisions which are based on the profile of the user, his/her preferences as well as on dynamically changing factors during the interaction process.

The adaptation manager specifies two different categories of adaptation properties. The first category is all the properties that drive the adaptation decisions which are inferred by the adaptation rules engine. Such properties can vary from the target user model characteristics and the preferences of the user, to dynamically changing properties such as the changing user position in relation to the position of the robot, the user standing or siting pose, his/her fatigue, or environmental factors such as the ambient room noise or lighting conditions. The second category of parameters, are

the actual adaptation parameters that change to reflect the changing of the properties of the first category. For example, when the ambient room lighting increases, the lighting parameter of the first category changes to reflect this change. The change in the lighting property has a result the triggering of the adaptation rule engine which infers that the contrast of the robot screen must be increased to match the change and make it more legible. As a result, the contrast adaptation parameter that belongs to the second category changes and this change signals an "adaptation needed" event.

The adaptation manager is based on the singleton software design pattern to ensure that only one instance of this component will be allowed to be active during the runtime. Furthermore, the adaptation manager contains the necessary bridging components between the framework and the FAmINE network environment in order to support augmented cognition scenarios based on events provided by the Ambient Intelligence environment. Furthermore, the necessary bridging components that connect the framework with the ROS operating system have been developed, to enable the adaptation manager to receive events from the robot operating system, concerning the calculated pose and position of the user that can be retrieved from the robot's backend.

```
when(UserIsStanding)
{
    ControlSizing = "BigSizes";
}

when(LightsAreOn && WearingGlasses)
{
    ColorScheme = "DarkColors";
}
when(!LightsAreOn && WearingGlasses)
{
    ColorScheme = "LightColors";
}

when(!WearingGlasses)
{
    ColorScheme = "HighContrastColors";
}

when(Distance == "short" && !UserIsStanding)
{
    ControlSizing = "SmallSizes";
}
```

```
when(Distance == "average" && !UserIsStanding)
{
    ControlSizing = "MediumSizes";
}

when(Distance == "long" && !UserIsStanding)
{
    ControlSizing = "BigSizes";
}
```

Figure 37: Sample ACTA adaptation script fragment.

The adaptation manager provides the necessary functionality for any control to register and unregister itself in order to receive the “adaptation required” events. To this end, the adaptation manager keeps a list of all registered components and notifies them whenever an adaptation parameter changes. In this case, the adaptation happens automatically without the registered controls taking any action. However, the adaptation manager provides the “adaptation needed” events as publicly emitted events so that custom developer components can take special action based on the adaptation parameters when adaptation is needed.

The adaptation manager has been developed to lazily load all the available styles from all three major categories and use them to achieve the desired adaptation. The loaded styles are kept in an internal dictionary that can be indexed using the different style names. This way, adaptation can be realized by a set of simple ACTA scripts that can be run in the internal instance of the adaptation manager’s rule engine.

shows a sample ACTA adaptation script fragment that consists of when statements. In this simple example, whenever the standing state of the user changes, the adaptation engine is triggered to reflect the sizing of the framework’s used controls. In addition, whenever the adaptation manager receives an event from the ambient intelligence environment that the ambient room lighting has changed, the active coloring scheme of the application changes to a more comfortable combination in respect to the end users.

5.5.2.3 *Adaptation strategies*

The developed adaptation manager of the proposed framework supports multiple types of adaptation for providing the necessary functionality for realizing different adaptation needs based on the dynamic nature of each supported scenario. In this

context, the adaptation manager has incorporated the necessary functionality to support static adaptation scenarios, dynamic adaptation scenarios, as well as adaptation scenarios that are driven by environmental changes.

5.5.2.3.1 Static adaptation

The static adaptation is the simplest and most straightforward type of adaptation. It is based on the virtual user models (VUM) of the target user population. The properties that drive this adaptation strategy include the gender of the user, his/her age, his/her expertise, the user's cognitive state, potential age-related disabilities and functional limitations, the user's communication preferences and modality personalization parameters etc. The adaptation manager incorporates all the necessary functionality for loading the user's profile and inferring the necessary adaptation decisions to present the user with a fully usable system in order to enable the interaction with it. The profile of the user is loaded dynamically from the VUM engine that runs on the linux backend of the robot and the user profile parameters are transferred to the proposed framework through the ROS interconnection.

5.5.2.3.2 Dynamic adaptation

The dynamic adaptation is based on dynamically changing factors during the interaction between the user and the robotic platform. For example, the platform might be outside the legible visual range of the user for him/her to be able to read the screen, or the user might have started interacting with the robot while sitting on a chair and then decided to stand up. Such dynamically changing factors reflect in the modalities selection of the robot as well as on the adaptation of each used modality. For example, if the user was in the middle of giving input to the robot through the virtual keyboard control when he/she decided to stand up, the virtual keyboard dialogue changes to display a bigger keyboard, with bigger margin and padding among the keys, a larger text preview field for the user to be able to read while standing and bigger buttons for accepting or rejecting any changes.

5.5.2.3.3 Adaptation through environmental sensing

The third type of supported adaptation is the adaptation achieved through environmental sensing. This type of adaptation relies on the ambient intelligence

environment in which the robot is deployed, to supply the robot with the desired information about environmental changing factors. For example, the environment can provide the robot with all the necessary environment status data including ambient lighting, ambient noise, the number of users in the house, potential emergency scenarios that take place beyond the sensing capabilities or sensing range of the robot and need attention, etc. In this scope, the robot becomes omnipresent in a sense due to its augmented cognition capabilities, while contributing to creating smarter robot behavioral models and functional supported skillsets. Furthermore, the increased capabilities of the robot that are based on the environmental sensing, contribute to higher levels of subjective user satisfaction, richer overall user experience and higher levels of acceptance of these technologies.

5.5.2.4 *User profiling and environment modeling mechanism*

The user profiling and environment modeling mechanism that is used for adaptation purposes in the context of the FIRMA framework's adaptivity and adaptability strategies, contains the necessary infrastructure to support adaptation based on the user profile (user model), the context of the interaction, as well as the context of the surrounding ambient intelligence environment. The current implementation of the mechanism supports one parameter from each one of the three different categories. It includes the user expertise parameter from the user modeling category, the user state (pose) from the interaction context category and the lighting state from the environment context category.

The profiling and modeling functionality that has been currently incorporated into the framework is preliminary since this component is used as a placeholder, in order to ensure that the framework has been designed properly to accommodate for a more complete profiling mechanism in the future.

Regarding the user profiling category, it supports only the user expertise parameter. According to this parameter, the time selection task is automatically parameterized and adapted to the user's expertise value. Novice and average expertise users are presented with different, more simple and intuitive controls and are guided through the time selection task, whereas expert user are given the capability of selecting the

desired time using a single, more complex dialogue. The user state (pose) and environmental state (lighting) have been also included in the profiling and modeling component. The user pose parameter can have either the "standing" or "seated" value which is reflected on the sizing adaptation of all framework elements, controls and dialogues. Finally, the value field of the environmental lighting parameter has the values "on" and "off" that get reflected on the runtime coloring scheme selection of the framework.

The existing profiling mechanism, though preliminary, can either be extended to include complete virtual user models and environment context ontologies that can drive the adaptability decisions of the framework or can be connected with already existing VUM components and profiling ontologies to load the necessary parameters.

In the context of the RAMCIP project, the VUM functional component will be implemented on the backend of the robot and will communicate with the FIRMA framework through the implemented ROS interface. Finally, the environmental context will derive from the sensors of the robots which will be made available to the framework through the ROS interconnection as well.

5.6 Globalization and Localization

Globalization is the process of designing and developing applications that function for multiple cultures. Localization is the process of customizing a given application for a given culture and locale.

The proposed framework provides inherent support for globalizing and localizing the developed applications to the native language of the users. The supported globalization functionality has been based on the automatic translation of all the framework user controls, dialogues and elements that are being used. The localization of the developed applications is realized by means of a universal translator auxiliary helper class that has been developed as part of the framework. The localization is based on localized culture and locale specific resource files that can be translated by either the developer or by expert Translators to the end user's native language.

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xml:lang="en-US"
                    xmlns:system="clr-namespace:System;assembly=mscorlib">

    <!-- Title and TTS for saying time -->
    <system:String x:Key="AlarmClockApplication.AlarmClock.ApplicationName">Alarm Clock</system:String>
    <system:String x:Key="AlarmClockApplication.AlarmClock.TheTimeIs">The time is, {0}</system:String>

    <!-- SCREEN: Clock -->
    <system:String x:Key="AlarmClockApplication.UCClock.TimeLabel">The time is:</system:String>
    <system:String x:Key="AlarmClockApplication.UCClock.ShowAlarmsButton">Alarms</system:String>
</ResourceDictionary>
```

Figure 38: Fragment from the translation file of the Alarm Clock Application for the en-US culture-locale.

Figure 38 shows a fragment of the translation file that was developed for the Alarm Clock Application and for the "en-US" culture and locale. The translation uses a resource file that specifies the corresponding culture and locale and uses the system namespace from the mscore library to declare literal strings that are going to be used as the translations when the different user interfaces are displayed in the respective locale. As shown in this figure, the translations are allowed to be format strings that are going to be used with numbered arguments for the creation of the final usable translation of the respective items. For example, the string that is used for the speech synthesis modality to synthesize the uttered prompt for the time of day, uses a placeholder for the actual time which is going to be filled in during the runtime, when the user asks the robot for the time.

One major point of the translation module is that all translations are based on keys which can be prefixed with any desired phrase. The translation mechanism was designed having in mind that each translated string should have a corresponding key which could be prefixed by the fully qualified name of the assembly that the translated element belongs to, followed by the name of the application that contains the element. However, when a translated control belongs to a specific application dialogue, the name of the respective dialogue is used instead of the application name. For example, as seen in Figure 38, the translation of the time label of the clock screen dialogue of the developed alarm clock application that displays the current time, is based on the key "TimeLabel" which is prefixed by the assembly name "AlarmClockApplication" followed by the dialogue name "UCClock". Moreover, translations for elements, controls and dialogues that are part of the FIRMA

framework, should be prefixed with the fully qualified name of the application's assembly name followed by the full application's name.

The translation process is automatically performed, based on the idea that each element or control that needs to be translated, has to provide at least a x:Uid or a x:Name attribute either from within the respective XAML description or from the code behind it. The "x:" prefix merely defines the xaml scheme of the windows framework extension WinFX. A complementary translation functionality that triggers the automatic translation of the respective user controls or dialogues is the implementation of the ITranslatable code interface. The ITranslatable code interface has been developed to include a translation function that uses a translator and a prefix. The user controls and dialogues that implement the ITranslatable interface are treated as black boxes by the framework. Having implemented the translation function, conveys that they are responsible for translating themselves in whatever way they seem fit. As a result, whenever the framework encounters an ITranslatable user control or dialogue during the translation process, it simply calls its translation function instead of going deeper the visual tree trying to discover any translatable elements. This feature has a twofold advantage. Firstly, whenever a user control or dialogue needs to translate specific elements of its contents, or needs to translate the contained elements under a specific prefix, it can just implement the ITranslatable code interface in order to have full control over the translation process. Secondly, by implementing the ITranslatable interface and then quietly taking no action inside the translation function, excludes the respective user control or dialogue from the translation process automatically.

```
<Button Grid.Column="0"
        Click="GoBack"
        VerticalAlignment="Top"
        Style="{DynamicResource TileButton}"
        awe:ButtonEyeCandy.Image="Images/back.png"
        Content="Back to Clock"
        x:Uid="BackButton"/>

<Button Margin="10" Grid.Row="1"
        Style="{DynamicResource KazNormalButton}"
        Click="ButtonClick"
        VerticalAlignment="Top">
    <TextBlock x:Uid="ShowAlarmsButton" Text="Show Alarms"/>
</Button>
```

Figure 39: Two different approaches to button automatic translation

Figure 39 shows an example of two different approaches regarding the translation of a button. The first button has no need to declare separate children and just defines its content as an XAML attribute named “Content”. It also specifies a x:Uid with the name “BackButton” which is used for its automatic translation in the context where it is used as a button. The second example shows that even if the button declares to have children, no matter how complicated their visual tree representation is, the translation process does not change as long as there is a text element (label or text block or implicitly created as child of a more complex control just like in the first example) that defines either a x:Name or a x:Uid. The second button explicitly defines to have a TextBlock as its child which declares a x:Uid with the name “ShowAlarmsButton”. As a result, it gets automatically translated during the translation process.

The translation process is automatically spawned by the RobotAppScreenBase base class which is the base class that every application dialogue based on the FIRMA framework must derive from. Furthermore, ecery application that is based on the FIRMA framework must derive from the RobotApp base class which provides the necessary translator engine instance to be used by the application dialogues. Figure 40 shows a snippet from the translation functionality of the RobotAppScreenBase. The translation process is automatically triggered from the initialized event of the corresponding application dialogue. Furthermore, it can be manually triggered from

inside the base class for providing the translation functionality to be used with the virtual keyboard input user dialogue which is automatically spawned from the auto-augmented text boxes, hence the user control that takes as an argument.

```
private void Translate(UserControl uc = null)
{
    if (uc == null) uc = this;
    UniversalTranslator translator = ParentApplication.Translator;
    DependencyObject depo = uc as DependencyObject;
    foreach (TextBlock t in depo.GetChildrenOfType<TextBlock>())
        t.Translate(translator, GetType().FullName);
    foreach (ContentControl cc in depo.GetChildrenOfType<ContentControl>())
        if (cc.Content is string)//This covers for Labels, Buttons, ComboBoxItems
            cc.Translate(translator, GetType().FullName);

    //...

    //Then, lets translate all ITranslatable components because they are blackboxes
    foreach (ITranslatable u in depo.GetChildrenOfType<ITranslatable>())
        u.Translate(translator, GetType().FullName);
}
```

Figure 40: Snippet from the translation functionality of the RobotAppScreenBase class

The translation process begins by retrieving the translator engine from the parent application that the respective dialogue is a child of. Then, a custom function that traverses the Visual Tree of the user control which is being translated is being called. This auxiliary function traverses the visual tree of the user control in search of framework elements that match the type of its argument. For example, the translation process first searches for all elements of the TextBlock type and then translates them using a set of developed auxiliary helper extension functions that know how to treat and translate each type properly. The translation process continues for various framework element types such as content controls, panels and containers in general. Finally, it does a last search for all ITranslatable items since they are treated as black boxes by the Visual Tree traversal function and are never “opened”. These black boxex implement the ITranslatable code interface and as a result “know” how to translate themselves properly, hence their implemented translation function is called with the prefix argument being the fully qualified name of the dialogue’s assembly.


```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xml:lang="el-GR"
                    xmlns:system="clr-namespace:System;assembly=mscorlib">
    <system:String x:Key="AlarmClockApplication.UCClock.TimeLabel">The time is:</system:String>
    <system:String x:Key="AlarmClockApplication.UCClock.ShowAlarmsButton">Show Alarms</system:String>
</ResourceDictionary>
```

Figure 41: Auto-generated missing translations. The UI that is missing some translations (top) which are prefixed with the hash marks and the automatically generated missing translations file (bottom), ready to be translated.

The whole translation process is based on the developed translation engine which runs as an instance of the UniversalTranslator class. The translation engine does not follow the singleton software design pattern, as the design decision was that each application should be responsible for managing its own translations which in turn should be kept in an instance of the universal translator class. The translation engine provides the necessary functionality for supporting different languages in terms of cultures and locales. The current set of supported languages includes Greek, English, Spanish, Catalan and Polish. The translation engine offers the functionality for setting and retrieving the current culture and locale, as well as the necessary infrastructure for loading and managing the translations of the currently selected language.

The translation as aforementioned is based on keys which can be prefixed with the context in which they are being used. The final translation is retrieved by joining the prefix with the key and looking it up in the dictionary of loaded translations. If the

translation of an item can't be found, the item is added to a pending translation list. This list contains the item key, its prefix as well as the development-time content of that item which is being provided to the translation engine by the auxiliary helper translation functions for the different framework types that were developed. As a result the translation engine has a complete overview of which items couldn't be translated both in terms of which was their keys and prefixes, and what was their development-time content. This is done so that all the items that couldn't be translated can be written to a file containing all the missing translations in order to facilitate the developers of new applications.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <appSettings>
        <add key="PlaintextGrammarsDir" value="plaintextGrammars"/>
        <add key="CompiledGrammarsDir" value="compiledGrammars"/>
        <add key="GENERATE_MISSING_TRANSLATIONS" value="true"/>

        <add key="LANGUAGE" value="greek"/>
        <add key="VOICE_GENDER" value="Male"/>

        <add key="ASRConfidenceThreshold" value="0.3f"/>

        <add key="ProxyContext" value="kazAmIProxy"/>

    </appSettings>
</configuration>
```

Figure 42: Sample application configuration file that defines the desired runtime language and triggers the automatic generation of missing translations.

For all missing translations, the development-time content of the elements is returned to the UI, prefixed with a hash mark. As a result, the developer has to keep in mind only the fact that each item that he wants to get translated, must provide either a x:Name or a x:Uid attribute. Then, at runtime, all the elements that couldn't be translated are collected into a missing translations file which contains elements ready to be included in the translation resources, in terms that they already have a key, a prefix and their sample development-time content. This takes place during the application shutdown after the first time it runs. Figure 41 (top) shows a sample UI running in the Greek language. The home button has been successfully translated, however the time label and the button are missing their translations and hence they

appear with their development-time content prefixed with hash marks. The bottom part of the figure shows the automatically generated translations file which is created during the application shutdown and contains all the missing translations along with their development-time content, ready to be translated and incorporated into the main translations file. The whole process is configurable from the applications configuration file as seen in Figure 42. Finally, the runtime language selection for the UIs can also be configured as seen in the same figure.

5.7 Transparent multi-application state management

The logic behind the orchestration of the dialogues of each application that is based in the FIRMA framework is that the application can be represented as a finite state machine (FSM). For simplicity purposes, each state of the FSM is mapped with a respective application dialogue, however multiple dialogues are allowed to be mapped to a single state or states are allowed to exist without a corresponding application dialogue. As a result, each application must be able of maintaining its state in respect to the FSM as well as in respect to the dialogue that is mapped with that state and has to be displayed on the onboard screen.

Since the robotic platform should have many applications that provide different functionalities that should be able to co-exist and run simultaneously, several problems had to be overcome. The need for functionality co-existence can be better understood by an example. For instance, the user might be watching TV and changing the channels using the onboard screen when the time for his medication intake comes. The robot should be capable of temporarily closing the television controlling application and then initiating the medication intake application that would remind the user that it is time to take the afternoon pills, while guiding him/her through the whole process of selecting the correct ones and validating that the user has taken them successfully. When the medication intake has finished, the robot should be capable of closing the medication application and re-opening the TV control application. Furthermore, the TV control application should be able to re-open displaying the same dialogue that was displaying when the interrupt took place to

minimize user frustration and increase user subjective satisfaction and acceptance of the robot.

The first problem with the above scenario was that the ACTA scripting language runtime was designed based on the singleton software design pattern, hence only one instance of the rule engine that could drive the FSM could exist. This had to be changed to allow for multiple simultaneous rule engine instances, one for each application. Furthermore, each rule engine should be optimized to evaluate only the rules that can affect the current state of the FSM in order to minimize the evaluation time during each iteration run and maximize performance. The process that was followed for allowing for multiple instances of the rule engine and the different optimization approaches that were followed is described in the section 5.2.

The next problem that was faced, was the fact that the applications should all define a starting state and they should all start in that state during the robot initialization. This was resolved by requiring each application to define a starting state in its constructor and pass it to the RobotApp base class constructor as well. The initialization of each application was assigned to the Main Menu application which is initialized and started by the main UI Navigator window. The main menu in turn, is responsible for scanning the applications directory for all the available dlls that are of type RobotApp, loading in an internal infrastructure and initializing them based on the starting states that each application defines. This way, all the applications start being correctly initialized and in their starting state, subject and eligible to receive and emit events according to their functionality.

The third major problem that was encountered is the functionality regarding the temporary pausing and resuming of applications while maintaining their state. Even more importantly, a prioritization mechanism was needed to manage and control which application should be allowed to interrupt the interaction process. This is a very important aspect, since the interaction process shouldn't be interrupted under certain circumstances. For example, when the user is interacting with the robot in the middle of an emergency scenario, the interaction process shouldn't be interrupted by any of the other applications for any reason. To overcome this problem, a prioritization mechanism was built, that is responsible for controlling the interruption priority of the

different applications which can be specified as priority rules in a predefined format. The whole functionality of this mechanism has been embedded into the FIRMA framework while the developed applications are being enforced to respect the prioritization rules by moving the control mechanism to the base class where all the applications derive from. The prioritization mechanism is discussed in 5.8.2.

Finally, the pausing and resuming of applications while maintaining their state was solved by not shutting down the paused applications but just removing them from the display instead. Each application is responsible for maintaining its own state as well as the knowledge of whether it has acquisition of the display resource at any given time. Furthermore, each application is responsible for internally storing its last displayed dialogue when it loses the privilege of displaying it on the screen. The application remains active in the background, receiving and emitting its respective events while it is in the pause state. The "pause" state refers only to the functionality of displaying application dialogues on the screen. However, if a paused application decides that it is essential to become active again and regain the display control in order to show some important piece of information, it can ask the communication planner for permission and if such permission is granted, the application is reactivated and granted access to the display again. Furthermore, if a previously paused application is reactivated by the user through the main menu, it gains access to the display and it re-displays its last displaying dialogue which had been internally stored to improve speed and performance. The time keeping capabilities of the applications that have been discussed in the previous sections are being paused and resumed accordingly when an application becomes inactive and when it becomes reactivated.

5.8 Enhanced human robot communication

5.8.1 Communication decision making

The communication decision making term refers to the communication decision that are being made by the robot regarding the selection, activation, deactivation and fusing of the available modalities. The available robot modalities include the speech synthesis component that can be used as an output modality, the speech synthesis

component that can be used as an input modality, the gesture recognition component that can be used as an input modality, and the touchscreen onboard the robotic platform which can be used both as an input and as an output modality for the user to give touch commands and receive feedback on the display.

The selection and fusion of the appropriate modalities at any given time was designed to be transparent to the developer because the different events and factors that drive the modality selection, activation and fusing are difficult to deterministically predict at design and development time. Moreover, taking into account all the possible combination among the factors that drive these decisions would increase the complexity of the developed applications exponentially as the developers would have to have manual control over the selection and manipulation of the different modalities for every developed application. This would lead to code redundancy and could introduce errors which could be difficult to reproduce and identify, since each developer would approach the modality integration process on his/her own. Leveraging the modality manipulation and management to the underlying framework solves the aforementioned problems since the developers are only responsible for using the modalities they desire and then the framework undertakes the delivery or the receiving of information through the respective channels. The developer is allowed to use any of the available modalities at design time and the framework refines this selection at runtime. For example the developer might choose to give both visual and auditory feedback for a user action by simultaneously using the screen for displaying information while using the text to speech interaction modality for delivering the same information over auditory cues. However, since the user might be in the context where the auditory feedback wouldn't be allowed at the time of interaction (either due to increased ambient noise or due to other restrictions, e.g., time of day), the framework can automatically decide not to deliver the auditory feedback but only use the onboard robot display instead.

Decoupling modality selection and fusing and the development of applications contributes to the increased scalability of the framework as additional modalities such as hardware buttons and switches can be included and incorporated with minimal changes. Since there is a central point for controlling which modality is allowed to be

active at any given time, the integration of additional modalities becomes a trivial process. For all added additional modalities, the developers would have to fine tune their applications in order to include them. However, already existing applications would be able to run without changes if they would need to incorporate the additional modalities.

5.8.2 Communication Planner

The communication planner module was developer to manage and orchestrate the different available modalities. This module is responsible for selecting, activating, deactivating and fusing all available modalities during the interaction process. The modality selection and fusing parameters that can drive decisions can come from different sources such as the backend of the robot, the surrounding environment, the time of day, the number of people in the room, the profile of the user, the state the user is in etc. The communication planner incorporates all the necessary functionality to accommodate these input sources. Furthermore, additional rules have been incorporated into the communication planner module that override any decisions made in cases where it is necessary. For example, the speech recognition input modality should never be used whenever the speech synthesis output modality is being used. As a result, the communication planner is responsible for initializing all the available modalities during the system startup process, hook up all available modality events and then use them to make such decisions. Whenever the speech synthesis output modality emits the “SpeechStarted” event, the communication planner should be able to detect it and automatically temporarily pause the speech recognition input modality to avoid scenarios where the robot might falsely recognize its own voice and take action on its own commands.

Apart from deciding over how and which modalities can be used at any given time during the interaction process, the communication planner has been commissioned with the role of deciding over the state traversal that takes place inside the different applications as well as granting or denying the applications requests for displaying information on the onboard screen. During to the nature of the communication planner module, it has been based on the singleton software design pattern to ensure that only one instance of this module is permitted to run during the runtime of the

system. The communication planner is able to receive requests from the different applications regarding either state changes of their internal FSMs or requesting permission to display information on the onboard screen. For the communication planner module to be able to respond to such requests, the currently displaying application state has to be recorded. To achieve this, the communication planner module stores the required information regarding which application is being displayed on the screen, which state that particular application is in, as well as which dialogue is that application displaying in that particular state. Furthermore, the communication planner module manages a set of permission rules that describe if an application that is requesting permission regarding a state change or a display request should be granted or denied this permission.

```
<?xml version="1.0" encoding="utf-8" ?>
<Permissions>
    <Permission FromApp="*"
                FromState="*"
                FromScreen="*"
                ToApp="MainMenuApplication.MainMenu"
                ToRequestType="*"
                ToStateOrScreen="*"
                Allowed ="true"/>
    <Permission FromApp="*"
                FromState="*"
                FromScreen="*"
                ToApp="AlarmClockApplication.AlarmClock"
                ToRequestType="SCREEN_DISPLAY"
                ToStateOrScreen="AddNewAlarm"
                Allowed ="false"/>
    <Permission FromApp="*"
                FromState="*"
                FromScreen="*"
                ToApp="*"
                ToRequestType="*"
                ToStateOrScreen="*"
                Allowed ="true"/>
</Permissions>
```

Figure 43: A fragment from the rules file that is being used by the communication planner module to decide over the grant or denial of permission to applications

Each rule is comprised of two triplets. The first contains the origin application name, state name and dialogue name and the second contains the destination application name, state name and dialogue name respectively. When a request is being made, the communication planner module searches the available rules in a linear fashion until it finds the first rule that matches the request. The rule is decided to match a request if and only if the origin triplet matches with the currently stored state of the

communication planner and the destination triplet matches with the request that has just been received. When the first matching rule is found, the permission is either granted or denied according to the rule's verdict. If no suitable rule is found that matches both the origin and destination triplets, the permission is denied by default. The star wildcard (*) can be used to denote that any application, state or dialogue name is eligible to be matched in the corresponding place where the wild card is being used.

An excerpt from a rules file that can be used with the communication planner module can be seen in Figure 43.

The figure shows a file containing three rules. The first rule permits the main menu application to gain access to the display, become active and display any of its available dialogues no matter what application is being currently displayed. This is to allow the home button that navigates the user to the main menu screen to be enabled at all times. No matter which application is active, when the user pushes the home button, the communication planner module grants the requested permission to the main menu application independently from the type of its request. The independence regarding the previously displaying application is denoted by the star wildcard characters in the origin triplet, while the independence in respect to the type of the main menu request is denoted by the wildcard characters in the request type and screen or state parameters in the destination triplet. In a similar way, the alarm clock application is always denied the permission to display the "AddNewAlarm" dialogue according to the second rule. Finally the third rule has been added to negate the default behavior of the communication planner module when no matching rule is found. Instead of denying the request, the third rule grants all requests from all origin states to any destination states if no other rule has been matched before. Due to the linear search mode of the rule file, the last rule will only run if and only if no other rule has been previously found to match the respective request. This way the behavior of the communication planner module can be completely customized to fit the needs of all the applications that have been deployed on the robotic platform. Finally, the rules are declared inside an XML file that can be loaded and reloaded at runtime to enable reconfiguration of the module on the fly during the runtime.

5.9 Augmented context-awareness through AmI environments (FAmINE)

An important role in the whole user robot interaction is played by the surrounding environment in which the robotic platform is deployed. Being able to receive information from the surrounding environment, the robot can use such intelligence to implement smarter interaction behaviors while providing a better and richer set of functionalities to the end users. The following sections discuss how inherent support for the FORTH's Ambient Intelligence Network Environment has been embedded into the framework, how the necessary mechanisms were built for realizing method calls and receiving events, as well as how FAmINE provides information can be used to achieve augmented reasoning capabilities.

5.9.1 Sensing through the environment

The FIRMA framework has embedded support for FORTH's Ambient Intelligence Network Environment. This was achieved by encapsulating the usage of the FAmINE middleware inside a singleton class that is a black box to the developers of applications. This class, namely the FamineWrapper, provides all the necessary functionality for resolving services, calling functions, retrieving properties and receiving events from FAmINE service while hiding the underlying code complexity to the end developers. This enables the developers to make smarter applications that take advantage of all the capabilities that such a smart environment has to offer.

The FamineWrapper blackbox was based on the singleton software design pattern to ensure that only one instance on the class will be available throughout the runtime of the system. This was done to conform to the design and programming guidelines of the FAmINE network environment that dictate that initialization and shutdown of the environment must take place only once during the runtime of a system. As a result, the initialization of the FAmINE environment happens during the lazy instantiation of the singleton class and the follow-up clean-up that is required during the system shutdown can be either explicitly requested or takes place automatically during the

shutdown process of the system since the class implements the IDisposable WPF interface according to its development guidelines.

The developers that want to use the intelligence that FAmINE can add to their applications, can resolve services simply by specifying three literal values, one for the file where the service description exists, one for the name of the desired service and one for the context under which the desired service should be found. The last value is to permit multiple instances of the same service to run under different contexts.

```
private dynamic ResolveService(FamineService fs)
{
    Type type = fs.Asm.GetType(fs.FullName);
    return typeof(Ami.Famine)
        .GetMethod("ResolveService", new Type[] { typeof(string) })
        .MakeGenericMethod(type)
        .Invoke(null, new object[] { fs.Context });
}
```

```
public object CallFunction(string serviceFullName, string method, object[] args)
{
    FamineService fs = _services[serviceFullName];
    Type type = fs.Asm.GetType(serviceFullName);
    MethodInfo methodInfo = type.GetMethod(method);
    return methodInfo.Invoke(fs.ResolvedServiceObject, args);
}
```

```
public event EventHandler<object[]> SwitchOnEventCommand;
private void SwitchOn_Executed(object sender, ExecutedRoutedEventArgs e)
{
    //this
    FamineBlackBox.FamineUtilities.Instance.CallFunction(service, "SwitchOn", new object[] { room, tv });
    //or this
    SwitchOnEventCommand.SafeRaise<EventHandler<object[]>>(d => d(this, new object[] { service, room, tv }));
}
```

Figure 44: Resolving a service(top), Calling a function inside the blackbox (middle) and calling a function at the developer's end using either the synchronous or the asychronous approach (bottom)

The resolving of the specified service from the specified context happens automatically based on reflection, using the “ResolveService” function that is offered by the FAmINE runtime. Having resolved a service, the developer can either call service functions, retrieve service properties or receive service events. All three are automated and each procedure is discussed below. Calling a function has been developed to be as easy as specifying the name of the service on which the respective function is desired to be called, the function name, and an array containing the function arguments. The calling of the specified function is achieved by reflection on the service assembly type. The function is retrieved from the assembly file of the

service, a MethodInfo object is created that contains all the desired method's metadata that are required to call it and finally the actual method is dynamically invoked on the resolved service object with the specified array of arguments returning its result to the caller.

```
IEnumerable<EventInfo> commandEventz = uc
    .GetType()
    .GetEvents()
    .Where<EventInfo>(ei => ei.Name.EndsWith("EventCommand"));

foreach (var item in commandEvents)
{
    string name = item.Name.Substring(0, item.Name.Length - "EventCommand".Length);
    MethodInfo addHandler = item.GetAddMethod();
    EventHandler<object[]> eh =
        (o, e) =>
        { FamineUtilities.Instance.CallFunction(e[0].ToString(), name, e.Skip(1).ToArray()); };
    Object[] addHandlerArgs = { eh };
    addHandler.Invoke(uc, addHandlerArgs);
}
```

Figure 45: Hooking up the asynchronous method calls with the actual methods that are being offered as part of a service

The developer has two options for calling a function, one synchronous and one asynchronous. The synchronous approach can be used when the developer needs the return value of the function and the asynchronous approach can be used when the return value of the function is not needed. The synchronous approach is discussed above where the developer specifies the name pf the service, the name of the desired method and provides an array containing the arguments. The asynchronous approach is a little more tricky as the developer has to declare an event of type "EventHandler<object[]>", named after the method he wants to call and postfixed by the term "EventCommand". Then, in order to call the desired function, the developer has just to raise the respective event and provide as the event arguments an array containing the name of the service and the argument of the desired method that is to be called. The FAmINE blackbox catches the emitted event, analyzes its name, looks up the desired function from the desired service and places the actual call using the specified array of arguments. However, this approach does not give the developer access to the returned value. Figure 44 shows the resolving of a service, the "behind

the scenes" calling of a function and the two approaches that are given to the end developers.

The asynchronous method call approach is based on the automatic hook-up of the developer created events to the actual methods that are being offered by the respective services. Figure 45 shows a fragment from the wiring procedure. The respective user dialogue is scanned to retrieve all the declared events whose names end in "EventCommand". Then, every discovered event is wired to an anonymous method that performs the actual method call whenever the respective event is emitted. The service on which the desired method will be called is the first argument inside the event's argument list, while the rest of the arguments are repacked into an array to become the arguments of the actual method call.

Receiving events from the ambient intelligence environment is even easier. All the developer has to do is to declare a method named after the event that is of interest, prefixed by the name "EventHandler_". The declared method must have the same number and type of arguments as the FAmINE event. Similarly to the aforementioned approach, the wiring between the developer defined method and the actual FAmINE offered events happens "behind the scenes" inside the FAmINE blackbox. Whenever an event is received, if the name matches the developer declared function, the provided function is called with the proper arguments so that the developer can handle the event appropriately.

```
public void EventHandler_TVVolumeChange(string room, string TV, int volume)
{
    if (room.Equals(this.room, StringComparison.InvariantCultureIgnoreCase) && TV.Equals(this.tv,
      StringComparison.InvariantCultureIgnoreCase))
        data.Volume = volume;
    //Refresh command bindings
    CommandManager.InvalidateRequerySuggested();
}
```

Figure 46: Receiving events from the ambient intelligence environment

Figure 46 shows a code snippet that demonstrates the implementation of the TVVolumeChange FAmINE event inside a custom developer dialogue. The event is prefixed with the "EventHandler_" literal and has three arguments. Whenever the

corresponding event is emitted by the ambient intelligence environment, the developer specified function is called and the event is handled accordingly. In contrast to the asynchronous method calling which was also based on events, this approach does not require the developer to specify the service from which the event will come because the FAmINE blackbox is able to search all available services and the respective offered events and make the connections automatically. The events are automatically disconnected from the developer's declared functions automatically whenever the respective dialogue gets unloaded from the screen.

5.9.2 Reasoning to achieve augmented context-awareness

Having all the necessary functionality to inherently support input from ambient environments available, opens new dimensions to the reasoning capabilities of the FIRMA framework. The framework has been given the capabilities to lookup and resolve services of the ambient environment, call methods, receive events and retrieve service properties. All this has made it possible for the framework to display higher levels of "intelligence" by incorporating different environment sensors and actuators to its reasoning modules.

```
// Author: Nick Kazepis <kazepis@ics.forth.gr>

#include <ami.idl>

module SampleService {
    interface AmIEnvProxy {
        // Types
        enum UserExperience { NOVICE, AVERAGE, EXPERT };

        // Methods
        boolean IsUserStanding ();
        boolean AreLightsOn ();
        UserExperience GetUserExperience();

        // Events
        void Event_UserIsStanding(in boolean isHeNow);
        void Event_LightsAreOn(in boolean areTheyReally);
        void Event_UserExperienceChanged(in UserExperience exp);
    };
};
```

Figure 47: Sample service that contains three methods and three events corresponding to three adaptation properties, one from each of the three major adaptation categories that are offered by the FIRMA framework

To demonstrate the aforementioned capabilities, a sample service has been developed that contains three methods and three events, each one of which corresponds to a values of each of the three different categories of adaptation properties. The first is a property concerning the user's skillset (experience), the second is a property regarding the dynamic position of the user in relation to the robot which has been discretized into the standing and sitting values and the third property corresponds to the status of the lights of the room the user is in. Figure 47 demonstrates the description of the developed sample service in the interface definition language that is used by CORBA which is the means on implementation for the FAmINE ambient intelligence environment.

The aforementioned service was implemented as a standalone application that run on a different distant machine in the same network as the sample application that was built based on the FIRMA framework. Figure 48 shows the implementation of the aforementioned service in the context of a standalone windows application. The "environment" was capable of informing the adaptation mechanism of the framework about changes in the state of the lights, changes in the position of the user (standing or sitting), changes in the user experience (however unrealistic, it was used to demonstrate the different instantiations of the adaptive component hierarchies) as well as some other information regarding the gender and the voice of the speech synthesis engine.

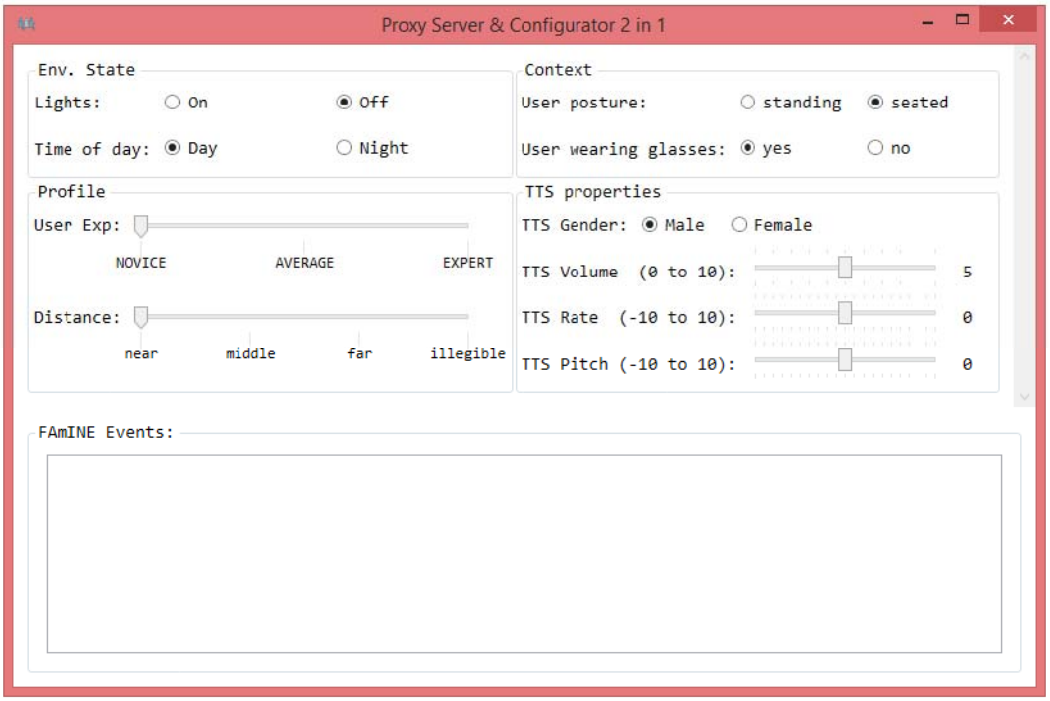


Figure 48: Instantiation of the sample FAmINE service that provides intelligence to the FIRMA framework through the ambient environment and the robotic platform's sensors

On the FIRMA framework's end, the received information was fed info the adaptation mechanism, namely the adaptation manager. The aforementioned properties were included into the properties that drive adaptation decisions and these properties were implemented according to the INotifyPropertyChanged component model interface. This was done in order to be able to notify the reasoning engine of the adaptation manager, whenever any of these properties changed its value that was triggered from an event coming from the ambient intelligence environment. A mapping between these adaptation driving properties and the actual adaptation values was performed and an ACTA script was developed to change the actual adaptation parameters accordingly. As a result, whenever the "environment" notified the adaptation manager that an adaptation driving property had been changed, the adaptation manager triggered the re-evaluation of the adaptation rules which came from the ACTA script and the actual adaptation parameters were changed accordingly to the facts that were inferred by the rule engine. Consequently, the "AdaptationSettingChanged" event was emitted by the adaptation manager and all the user controls, components and application dialogues that had been registered

with the adaptation manager to receive these adaptation events, were notified about the change and refreshed their adaptation parameters to retrieve the new values.

Incorporating the ambient intelligence environment into the adaptation mechanism of the FIRMA framework, offers new possibilities towards building smarter robots, enriching their functionality by enriching the sensors of the ambient intelligence environment that the robotic platform is deployed into and as a result, providing support for building more complex scenarios where the robot becomes part of the ambient intelligence environment.

5.10 RoboThink: A reasoning ruleset for FIRMA based assistive robot applications

The FIRMA framework is targeted to developers whose objective is to easily and effectively build elderly-friendly multimodal interactive applications for assistive robots. To this end, the framework comes with a pre-installed reasoning ruleset that can be used for adaptation purposes in order to be able to infer facts about some fundamental adaptation decisions when the robotic assistants come into play. In this scope, the developer is given the opportunity to enable this available rule library to activate the contained rules that provide adaptation for various aspects of the developed applications in the context of different situations, based on a number of predefined adaptation properties. The respective rules are provided as a separate rule file that can be automatically loaded in the adaptation manager functional component in order to drive the corresponding adaptation actions based on the selected properties. The provided ruleset provides ready-made adaptation for the following scenarios:

- Whenever the robotic platform is moving, the screen becomes unresponsive to touch events, based on the assumption that since the robot is moving, any touches on the screen are more likely to cause unintentional triggering of tasks than provide desired actions

- Whenever the robot is moving its hand, the screen gets deactivated in a similar manner as when the robot is moving while providing to the user only the functionality for an emergency stop of the corresponding action
- Whenever the user is supposed to interact with the robot but the robot is out of touch range, the robot (if possible) moves to come closer to the user so that he/she will be given the option to use the touch modality
- Whenever the developer marks a decision dialogue as important, the robot makes sure that the user sees and understands it and responds accordingly. To achieve this, the robot moves and turns itself towards to the user, highlights the important decision with high contrast colors and asks the user for confirmation. In this scope, auto dismissible notifications are automatically converted into manually dismissible notifications while the robot keeps asking the user for confirmation while the notification is displayed on the screen. For important dialogues, the voice input modality is deactivated and only the touch input and gesture recognition modalities become eligible for use to minimize false positives.
- Whenever the robot starts interacting with the user, the displayed fonts, framework elements and application components get automatically sized according to the distance between the robot and the user and the time of the day (bigger fonts and elements at night for more convenient interaction).
- Whenever the robot provides audible feedback to the user, the speech synthesis and audio volume get adjusted according to the distance between the robot and the user. The further the robot is away from the user, the higher the volume is in order to make sure that the information can be conveyed.
- Finally, whenever the robot gets too far away from the user so that the screen becomes illegible, the contents of the screen get replaced by a full screen photo that corresponds to the current action or task of the robot. For example, if the robot is in the middle of housekeeping (removing fallen objects from the floor) the screen changes to display a full screen broomstick.

5.11 Integration with ROS

In order for the framework to be able to interoperate with the robot operating system which runs at the backend of the robotic platform, the implementation of a ROS client library for windows was necessary. The implementation of the ROS client library for windows in the C# language was an effort started by Eric McCann, a research assistant at the University of Massachusetts Lowell Robotics Lab and Mikhail Medvedev (IBM, Austin, TX) under the name ROS.NET[14].

ROS.NET is a series of C# projects and one C++ project (a p/invoke wrapper around XMLRPC++) that allow a managed .NET application to communicate with any other ROS nodes that may be running on the backend of the robot on some other linux machine. To use it, the ROS_HOSTNAME and the ROS_MASTER_URI environment variables must be set to contain the IP of the managed side application computer and the Linux machine that runs the master node respectively. The ROS.NET managed library is partly an implementation of the ROS client library for the managed .NET framework along with some hacks to make the communication possible. For example the message generation process has been automated to happen autonomously during the subprojects building phase by respecting the folder structure that lies under the "messages" folder. This enables and supports references that use just the desired referenced message name instead of the entire messages subproject. Furthermore the developed library:

- includes support for the implementation of dynamic reconfiguration and all the prerequisites that are needed
- can generate a C# messages DLL containing standard, as well as custom message classes that will match MD5s (99% of the time) with and successfully send and receive (100% of the time when md5s match) to and from ROS nodes in officially supported ROS client languages
- Allows a nearly ROSCPP API for all of the familiar ROS programming elements such as publishers, subscribers, rosparam, service clients and servers, etc.

[14] https://github.com/uml-robotics/ROS.NET

The developed ROS client library has been used by its authors to communicate between various WPF windows and robots for a handful of research projects, and in their words "it could not have easily been replaced".

Although the ROS.NET client library provided sufficient functionality to serve as a stepping stone for integrating the ROS operating system with the FIRMA framework, several communication aspects needed to be taken care of. Furthermore, the approach towards creating managed publishers and subscribers had to be re-implemented to follow the application lifecycles as they are defined in the WPF control lifecycle design guidelines to ensure the maximum compatibility between the framework and the ROS nodes that had to be developed for the framework's communicational needs. Finally, the service publication and subscription mechanism had to be inspected and debugged as this part had not been tested by the original authors of the ROS.NET library and caused several errors. In particular, the client library was unable to successfully resolve some custom publishers and services, and the unregistration of nodes during their shutdown did not succeed leaving ghost nodes that caused pollution to the ROS middleware.

In order to realize and implement the aforementioned changes without interfering with the original managed .NET library and in order to make it easy to merge the changes back to the original master branch, a fork of the entire codebase was performed to a local repository where the necessary adjustments, changes and additions could be easily made. The local repository started containing a copy of the original ROS.NET client library and was then modified accordingly to tweak and implement the necessary functionality. The forked repository[15] was hosted in github under another account in order to be able to create a pull request when the changes were implemented and the resulting code base was ready to be merged back to the original library. Since the necessary changes concerned the fixing of bugs and the sanitization of the publication and subscription mechanisms to conform to the design guidelines of the WPF framework, their integration to the original code base was a very important step.

[15] https://github.com/nickkazepis/ROS.NET

The resulting ROS client library incorporated all the necessary functionality for creating and consuming ROS services and sending and receiving ROS messages. The managed library offered the capability of creating new ROS nodes, advertising publishers based on standard and custom ROS messages, subscribing to topics that used both standard and custom ROS messages, advertising services based on both standard and custom message definitions and finally consuming ROS services that were defined in the robot's backend linux machine. Furthermore, all the errors regarding the resolving of topics and services were fixed, and the unsuccessful unregistration of nodes was corrected.

The setup on which the aforementioned capabilities were tested, was a virtual machine running Ubuntu 14.04 LTS and the ROS indigo flavor. Custom messages and service types were developed to test the respective functionalities. The virtual machine had its own IP address and run the ROS master service. The ROS_MASTER_URI variable was pointed to the IP of the virtual machine running the ROS master, while the ROS_HOSTNAME variable was pointed to the windows machine IP which was running the sample windows applications. The communication between the two machines was stable and robust throughout the tests even in scenarios where multiple publishers, subscribers and services were active at both sides while any overhead on the system appeared to be constant with no fluctuations in processor or memory usage.

```
#region Vector Length Service
private ServiceServer simpleServer;
private DateTime lastServed;
private bool VectorLength(VectorLength.Request req, ref VectorLength.Response resp)
{
    DateTime now = DateTime.Now;
1   resp.length = Math.Sqrt(Math.Pow(req.x, 2) + Math.Pow(req.y, 2) + Math.Pow(req.z, 2));

    Dispatcher.BeginInvoke(
        new WriteToOutputWindowDelegate(WriteToOutputWindow),
        new object[] {
            now.ToLongTimeString() + ": Just calculated: sqrt(" + req.x + "^2 + " +
            req.y + "^2 + " + req.z + "^2 = " +
            resp.length + " [" + String.Format("{0:N2}",
                (now - lastServed).TotalMilliseconds) + "]" });

    lastServed = now;
    return true;
}
#endregion
```

Figure 49: The implementation of a ROS service on the windows side, using the ROS.NET client library for C#.

The developed ROS client library is used to interconnect the FIRMA framework with the robot's ROS backend. This way the framework to be able to send and receive commands (service calls) while sending and receiving messages as well. This is of utmost importance in order not only to incorporate the robot's (and to some extent the surrounding ambient environment's) sensors to the frameworks reasoning process but to be able to control the actual hardware of the robot as well. The ROS interconnection has been offered as the capability to create ROS nodes and manipulate the already existing ones in the backend.

```
private NodeHandle simpleNodeHandle;
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    //initialize ROS
    ROS.Init(Environment.GetCommandLineArgs(), "winnode_wpf");
    //create our node
    simpleNodeHandle = new NodeHandle();
    //initialize a listener on the /chatter topic
    simpleSubscriber = simpleNodeHandle
        .subscribe<Messages.std_msgs.String>("/chatter", 100, subCallback);
    //initialize a custom service
2   simpleServer = simpleNodeHandle
        .advertiseService<VectorLength.Request, VectorLength.Response>(
            "/vector_length",
            VectorLength
        );
    //initialize all timestamps
    lastReceived = lastPublishedTimestamp = lastServed =
        lastRequestedTimestamp = DateTime.Now;
    //initialize the publishers' thread
    publisherThread = new Thread(new ThreadStart(PublisherWork));
    //initialize the client thread
    clientThread = new Thread(new ThreadStart(ServiceClientWork));
    //start the publishers and the client thread
    IsRunning = true;
    publisherThread.Start();
    clientThread.Start();
}
```

Figure 50: The advertisement of a ROS service on the windows side, using the ROS.NET client library for C#.

The developer of an application can easily create a new ROS node, resolve a service that is running on the robot's backend on a specified topic and call the desired service functions to instruct the robot to take action. This is the part that makes the whole robot interactive and not just the user interface. Furthermore, the adaptation manager was given the functionality to create a ROS node and subscribe to the sensor publishers that run on the backend in order to receive notifications about dynamically changing parameters regarding both the user and the surrounding environment. This has given the framework augmented reasoning capabilities that can drive adaptation to new levels that cause increased usability, user satisfaction and acceptance of these technologies while contributing to a richer user experience.

```
#include "ros/ros.h"
#include "beginner_tutorials/AddTwoInts.h"

bool add(beginner_tutorials::AddTwoInts::Request  &req,
         beginner_tutorials::AddTwoInts::Response &res)
{
  res.sum = req.a + req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: [%ld]", (long int)res.sum);
  return true;
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "add_two_ints_server");
  ros::NodeHandle n;

  ros::ServiceServer service = n.advertiseService("add_two_ints", add);
  ROS_INFO("Ready to add two ints.");
  ros::spin();

  return 0;
}
```

1

2

Figure 51: The implemetation and advertisement of a ROS service using the official roscpp client library

Figure 49 and Figure 50 demonstrate the implementation of a simple ROS service (that calculates the length of a vector) at the Windows side using the ROS.NET client library for C# as well as its advertisement to the ROS operating system in order to be able to be used from the Linux backend. Furthermore, Figure 51 demonstrated the same approach using the original roscpp client library for implementing and advertising the same service on the backend Linux side. By comparing and contrasting the three pictures, the similarities between the two client libraries can be observed. The developers can develop ROS nodes, publishers, subscribers, services and clients in the same way that would use the original roscpp client library to develop them for the Linux backend. Figure 49 (1) is the exact equivalent of Figure 51 (1) for manipulating the response of a ROS service, while Figure 50 (2) is the exact equivalent of Figure 51 (2) for advertising the created ROS service to the ROS operating system so that it can be discoverable and callable from the Linux backend.

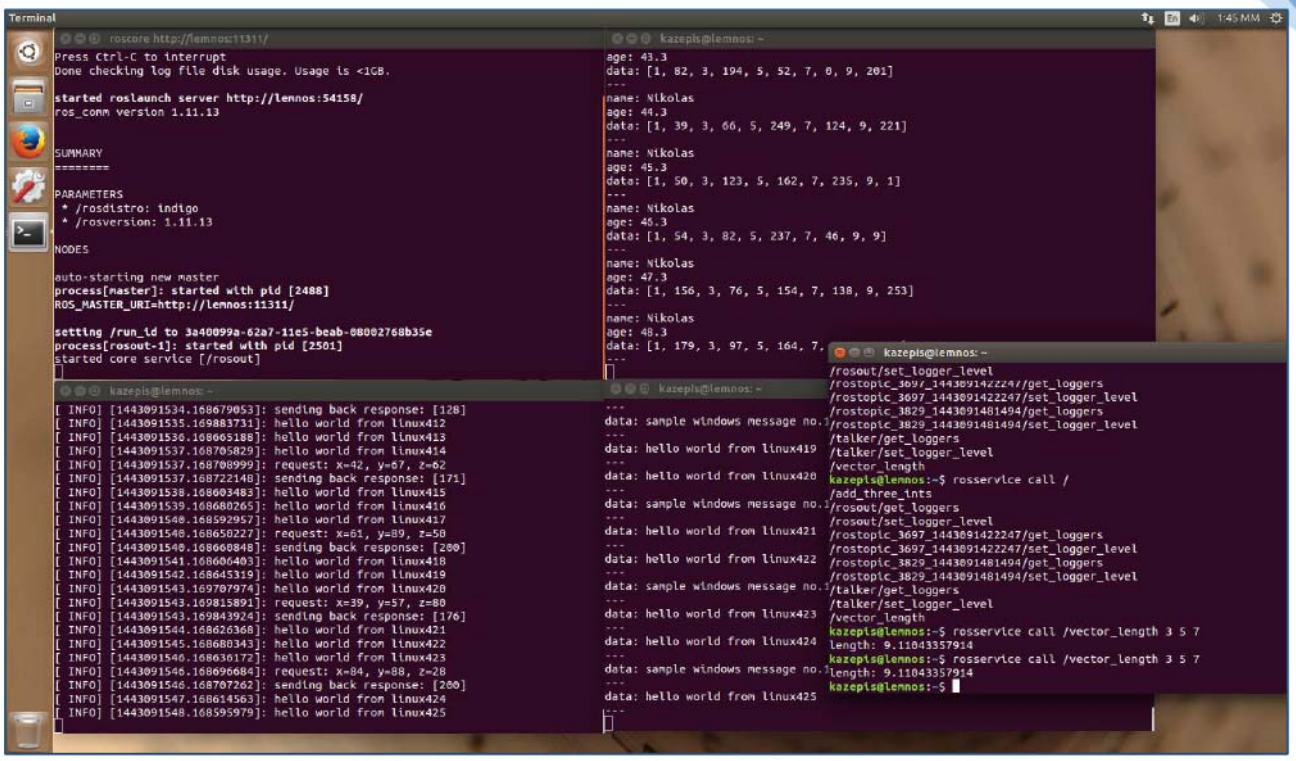


Figure 52: A virtual machine runnning the roscore and various publishers, subscribers and services

Figure 52 shows a virtual machine running the roscore prerequisites for the ROS operating system while several terminals run publishers, subscribers and services. The virtual machine is able to send and receive messages from the windows machine where the windows version of the ROS client library runs. Furthermore, the Linux side is able to receive service calls from the windows side and respond accordingly as well as invoke remote service calls on the windows side and receive the respective answers successfully.

Figure 53 shows the windows WPF application that connects to the backend ROS instance, subscribes to its messages, publishes its own custom messages to the Linux backend, receives and serves service call from the Linux side as well as successfully calls remote services that lie on the Linux side and receive the respective answers. This application is a windows application that comprises a WPF window and a textbox that has been styled to look like a console window. However the full range of graphical WPF capabilities are available on the windows side.

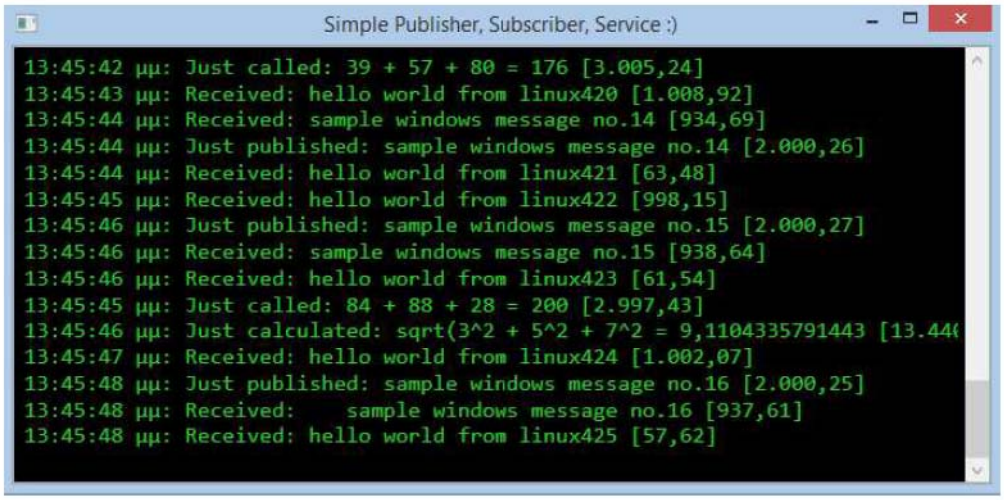


Figure 53: Windows application running reveral ros publishers, subsribers and services on the windows side

Finally, all the changes were packed into a pull request at the original ROS client managed library ROS.NET and was merged into the master branch a few days later by the original authors or the project.

5.12 A common intelligence: The user, the robot and the environment

The users, the domestic robotic platforms and the ambient intelligence environments in which both of them exist and interact form a very special triad. On the one side, there is the user with his particular needs, his preferences and his potential disabilities or impairments, who needs to be able to interact with the robotic platform and the environment. On the other side, there is a domestic robotic platform which is deployed into an ambient intelligence environment, capable of exploiting its potential to the maximum, having as its sole purpose to serve its user as best it can. Furthermore, the objective of the robot is to make its user feel enabled to live independently and safely at home rather than proactively overtaking many of his daily tasks and chores around the house. The user wants to be able to use the robot and the intelligent environment as a tool and a means towards achieving a better quality of life rather than feeling replaced and overruled inside his own home.

The robot's role is very tricky to implement, as the very delicate equilibrium between the empowerment of the user and the provision of proactive services and functionality has to be maintained. If the robots take too much initiative, the user might feel marginalized in his own life, however if the robot is totally dependent on the user it might be perceived as an unwanted redundant burden. On the other hand, the robot should be able to exploit the intelligence offered by the surrounding environment as much as possible in order to provide the user with valuable functional services. This can lead to increased levels of user satisfaction as the robot will be seen as an invaluable tool, as a caregiver, as an irreplaceable companion and more importantly as a friend and as a guardian that will always be there whenever the user needs it. By incorporating the ambient intelligence of the environment into the robot's reasoning logic, the robot can have knowledge about everything that is going on inside the house, including the physical state of the user. As a result, the user is protected against life threatening incidents such as appliance related accidents (which the robot can "foresee" and prevent) or health emergency situations that the robot can detect and instantaneously call for help.

Furthermore, the robot's intelligence contributes to a smoother learning curve as the user is not required to adapt to the robot's interaction patterns but the robot adapts to fit the needs and preferences of the user to the extent where the interaction between them feels natural. By incorporating all the different available modalities at the disposal of the user, the interaction process becomes indistinguishable from the daily life tasks that the user is accustomed to. The user does not need to remember how to operate the robot. He can simply talk to it or wave at it and the robot is smart enough to infer the user's intentions and act accordingly. Furthermore, the user feels safe and protected because he/she knows that whatever happens, the robot will be there to support him independently if the robot is in the same room as the user or any other room. In the former case the robot would be able to sense the emergency, whereas in the latter situation the smart environment would convey the required information to the robot.

However, carefully designing and planning the autonomy levels of the robot as well as the degree of its proactiveness is of utmost importance, since it directly affects and

influences the levels of acceptance that the robotic assistant will achieve. If the robot and the environment are relatively subdued and do not actively engage with the user, they will be considered redundant and useless. On the other hand, if the robot's behavior is too pervasive, it will be easily perceived as an intruder to the user's life and become rejected. Striking the golden mean without yielding to either of these pitfalls can be a very interesting and challenging problem to solve.

This work tries to contribute in this ongoing struggle by providing the means to developers for developing adaptive multimodal applications for robotic platforms without having to mess with the mechanics of multimodality and adaptation while providing the best to the end users from an HRI perspective along with ensuring the optimum cooperation and interconnection of applications with the robotic platform itself and the smart environment hosting it. As a result, the behavior of the robot can be configured and tweaked accordingly, to provide optimal levels of engagements with the user and proactiveness towards assisting his needs.

6 Evaluation and Validation

6.1 Developer based Evaluation

Developer based evaluation refers to the evaluation of the FIRMA framework by application developers. These developers can potentially be users who:

- Write code which extends but does not change the framework/software, e.g., a client to some service endpoints, a pluggable component coded against some extensibility point, or a client software application that uses the proposed framework to provide elderly-friendly multimodal interaction. As an analogy, a software application developer of a private company who has bought the FIRMA framework to create elderly-friendly interactive multimodal applications.
- Write code that changes the framework/software, e.g., fixes bugs, makes the software more efficient, or extends its functionality. As an analogy, a software engineer who has access to some of the framework's components' source code such as the ARMA runtime component and maintains or extends them.
- Are project members and have write access to the source code repository. Unlike the above two categories, these developers have to be aware of such issues as what the policy is on upgrading to use new versions of prerequisite packages, coding standards, who owns copyright, licensing, how changes are managed, if they're expected to support components they develop, how the project is run etc. As an analogy, the author and main developer of the FIRMA framework.

6.1.1 Methodology followed

The method that was selected for the developer based evaluation is the tutorial-based assessment approach [144]. Tutorial-based evaluation provides a pragmatic evaluation of usability of the software in the form of a reproducible record of experiences through the use of scenarios. This gives a practical insight into how the software is approached on a developer basis and helps detect any potential technical barriers that prevent adoption.

Tutorial-based evaluation reflects the experience of installing, configuring, using, learning and building/creating using the software that is being evaluated. The result is a forthright honest report based on the developers' experiences, explaining what was achieved with the software, including potential issues and shortcomings observed as well as suggested workarounds and fixes for these issues. The evaluation revolves around carrying out typical tasks using the software. In our case, the "software" was the FIRMA framework and the selection of tasks was based on them being representative of the FIRMA framework core functionality and intended purpose, including getting the framework ready to be used, e.g., downloading and installing the necessary components, setting up a development environment and making the framework components available for development.

This decision was based on the consideration that there are no available heuristics or guidelines for evaluating a development framework from the perspective of the developers. Additionally, the FIRMA framework is a unique tool, i.e., there are no known existing frameworks providing equivalent functionality that can be used for comparative assessment. These two facts have ruled out the use of non-empirical evaluation methods. On the other hand, the existence of a high fidelity prototype made feasible the use of an expert user-based method, i.e., the developers. The available options were to measure user performance, as well as to assess the users' subjective opinion on the perceived usability of the framework in terms of creating interactive multimodal elderly-friendly applications. Given the primary intention of the current effort at the time (to demonstrate the technical feasibility for such a tool before developing a commercial product), the subjective measurement was selected as more informative of the opinion of developers.

Additionally, given the target user group of the tool, i.e., developers, who are by definition expert users, it was decided to combine user satisfaction measurement with expert user interface evaluation in order to obtain detailed developers' comments and suggestions on the FIRMA development framework as well as its interface design regarding the ready-made components and framework elements.

The IBM Usability Satisfaction Questionnaires [145] was adopted for subjective usability measurement in a scenario setting, and namely the After-Scenario

Questionnaire (ASQ), which is filled in by each participant at the end of each scenario, and the Computer System Usability Questionnaire (CSUQ), which is filled in at the end of the evaluation. These questionnaires are available for public use, have been satisfactorily in use for several years now and are considered as extremely reliable. The questionnaires are reported in Appendix VII.

The result of the subjective evaluation with the IBM Computer Usability Satisfaction Questionnaires is a set of metrics which can be summarized as follows:

- ASQ metric provides an indication of a participant's satisfaction with the system for a given scenario;
- OVERALL metric provides an indication of the overall satisfaction score;
- SYSUSE metric provides an indication of the system's usefulness;
- INFOQUAL metric is the score for information quality;
- INTERQUAL metric is the score for interface quality.

6.1.2 Evaluation Procedure

The FIRMA framework was evaluated by six expert users with substantial experience in application development. All users had at least a University degree in Computer Science or related subject. All of them had at least a few years' experience in the field of creating WPF applications using the C# programming language and some basic knowledge, but no extensive experience or practice, concerning adaptation or localization practices or multimodality approaches. The user group consisted of four males and two females, whose age ranged from twenty-five to thirty-five years.

The group of users was briefly introduced to the main objectives of the FIRMA framework and of the evaluation experiments, and was provided with the following material:

- a brief introduction to the setup of the development environment;
- a brief description of the FIRMA framework's functionality and tools;
- a brief scenario (including an accompanying tutorial) involving the creation of a new toy application that consisted of two dialogue screens, as well as the integration of different modalities, adaptive tasks, adaptation and localization,

in order for the developers to be able to perform a more extensive testing of the system's features.

The developers were then requested to perform the tasks in the scenario, and fill-in the user satisfaction questionnaires, as well as an expert evaluation report as detailed as possible.

6.1.3 Evaluation Results

The results of the user satisfaction measurement are reported in Table 3 (ASQ) and Table 4 (CSQU).

	User 1	User 2	User 3	User 4	User 5	User 6	Average
Scenario A "Create a basic new application"	2,3	2,5	2,1	2,2	2,3	2,1	2,25
Scenario B "Integrate multimodality, localize it and add restrictions"	3,7	3,2	3,6	4,0	3,6	3,5	3,60

Table 3: After-Scenario Questionnaire (ASQ) Results (Range from 1 - highest - to 7)

Scenario A showed a variance of 0,019 which resulted into a standard deviation of σ=0,1384 while scenario B showed a variance of 0,056 which resulted into a standard deviation of σ=0,2380.

	User 1	User 2	User 3	User 4	User 5	User 6	Average
SYSUSE	2,2	2,3	2,3	3,0	2,1	2,1	2,33
INFOQUAL	3,3	3,2	3,3	3,1	3,6	3,6	3,35
INTERQUAL	3,0	3,1	2,0	2,8	2,0	1,2	2,35
OVERALL	3,1	2,2	2,5	3,0	2,5	3,0	2,72

Table 4: Computer System Usability Questionnaire (CSUQ) Results

The conduct of the specific "Create a basic new application" scenario appears from the results to have been easier than the conduct of the "Integrate multimodality, localize it and add restrictions" scenario. This is probably due to the need of developers to acquire some experience in how the framework works, what functionality it offers and how this functionality can be achieved.

The most appreciated aspects of the system were found to be its ease of use and overall effectiveness in the context of multimodality integration, automatic adaptation and localization and the reflection of the appearance of the end result during the design and development time in the context of creating elderly-friendly interactive multimodal applications. The users found the required workflow for the creation of new apps to be pleasant and intuitive and they were pleasantly surprised by the different supported automations that were supported by the system "out of the box" such as the multimodality integration, adaptation and localization processes. Concerning the included user interfaces and dialogues of the FIRMA framework, the users found that they are self-explaining, and that the dialogue screens do not contain information that is irrelevant. Font choices, colors, and sizes were considered consistent, and icons and other graphical elements were considered intuitive. The users also appreciated the fact that the framework is carefully designed to prevent common problems from occurring in the first place (such as the automatic inclusion of the design time style sheets which reflect the appearance of the end product), and makes dialogues, actions, and dependencies visible. The developers particularly liked the decoupling between the application dialogues and the application logic and were enthusiastic about the fact that fine tuning of the application logic can be done at a higher level without requiring the recompilation of the entire application code. Error messages were also considered to be clear and precisely indicating the problem at hand. A proposed improvement was to include in the dialogues (e.g., in the missing state checking feedback dialogue) suggestions for correcting errors. Furthermore the users offered helpful comments towards enhancements which are discussed later in this section.

The identified weak points of the framework mainly concerned the limited documentation provided. This was a known shortcoming of the prototype system, attributed to existing constraints at development time, leading to rather limited and focused documentation. The provided tutorial and documentation was focused on the parts of the workflow at which the developers were expected to have the least experience.

As already mentioned, the developers were also requested to provide an expert evaluation report accompanying the filled-in questionnaires. In these reports, the users offered their overall comments as well as more detailed suggestions for improvement of the FIRMA framework. The overall attitude of the users towards the system was positive. It was also pointed out that the tool presents a low cognitive load, and employs workflows and concepts familiar to application developers. However it was also observed that that developer had to maintain and meddle with different technologies to make an application work, something that was an expected forthcoming stemming from the nature of multimodally enabled applications.

The developers pointed out that there are some parts of the workflow that could be made further error-proof by providing some additional tools and editors. For example, the developers found the ACTA scripting language rather enjoyable but almost all of them commented that they would like some kind of auto-completion and some code snippets that could expand to provide some skeleton code for creating an additional application state or a transition between the current state and the rest of the states of the current application. Furthermore, they pointed out that the translation of the ACTA scripts into windows workflow foundation rulesets should be something that should be addressed by the framework itself automatically to avoid synchronization error between the rulesets and the source script files. This was a unanimous request which was already in the current work's future development list. Moreover, the developers suggested that other parts of the workflow such as the creation of SRGS grammars or the creation of restriction rules in the Communication Planner required adding code in XML which was not very convenient for all of them in respect to their experience with the language. In particular, the majority of them suggested that an SRGS editor should be provided to minimize user errors during the creation or the localization of SRGS grammars. Furthermore the development of an additional editor was advised towards supporting the creation of restriction rules for the Communication Planner while taking advantage of the semantics of the ACTA language which could provide automatic listing of all the available states, dialogue names and transition triggers. In addition, the developers suggested the creation of automation projects for the required project types, class types and dialogue types in

the Visual Studio IDE which was a foreseen request since the developers of the actual prototype system had already contemplated on the provision of such functionality in a subsequent version. Other comments concerned limitations and bugs of the current implementation (e.g., window resizing problems, lack of some confirmation dialogues, etc).

In general, the developers stressed that the availability of such a framework would in their opinion be very helpful in creating elderly-friendly multimodal interactive applications easily and effectively. However, it was also noted that a certain degree of familiarity with the framework needs to be acquired before effective use in real development cases, particularly in relation to the order of the tasks that the user has to perform, which may not be clear at a first glance. A useful suggestion put forward by four of the users was the addition of a step by step wizard for better guiding the novice users of the framework, acting as a tutorial for understanding the framework's functionality. In general, the need of easily searchable support information, for example in the form of Frequently Asked Questions, and of instructional interactions was pointed out. Furthermore, some of the users had specific requests for additional functionality and system capabilities they would like to see supported in future versions of the framework. These mainly concerned the inclusion of additional modalities, the formalization of the communication protocol between the framework and the ROS operating system, the provision of automation class types in the Visual Studio IDE for creating ROS nodes and subscribers and expandable code snippets for adding application dialogues in code behind.

In general, the user evaluation of the FIRMA framework offered valuable insights into the functional and the interaction characteristics of the system and reinforced the belief that there is an actual need and demand for a framework providing the building-blocks and tools to support the design and development of elderly-friendly interactive multimodal applications for assistive robots.

6.2 FIRMA vs. Visual Studio® alone

The aim of this section is to elucidate the benefits of employing the FIRMA toolkit instead of the Microsoft's Visual Studio alone, in terms of the developer's

performance and efficiency. This is achieved by applying simple source code development metrics (lines of code) to measure the reduction of code. The Visual Studio Lines of Code metric indicates the approximate number of lines in the code. The count is based on the IL code and is therefore not the exact number of lines in the source code file. Code metrics, for the most part, ignores generated code since the majority of that code typically can't be modified by the developer directly. Furthermore metrics results for an anonymous method that is declared in a member, such as a method or accessor, are associated with the member that declares the method. They are not associated with the member that calls the method. Hence the total lines of code metric, as it is offered directly from the visual studio tools is a valid metric to measure the reduction of code using the FIRMA framework.

In this section the development of the alarm clock application will be presented, taking into account the appropriate integration and coding steps required for a developer to build such an application from scratch. The same application will be revisited in an FIRMA oriented development scenario where the same developer employs the FIRMA framework to rapidly prototype the same application. Based on the presented workflow in this chapter simple metrics are applied to the two alternative development workflows so as to assess the overall reduction of effort that is achieved via the usage of FIRMA.

6.2.1 Typical development stages

In order to build a fully functional alarm clock application, the developer would need to follow the following steps:

- Develop the different interaction modalities including their parameterization mechanisms in order to make them adaptable and adaptive
    - Develop a speech synthesis modality which should be able to select at runtime the gender, pitch, rate and volume of the voice
    - Develop a speech recognition modality which should be able to compile, load and unload speech SRGS grammars at runtime, handle literal and semantic recognition results, handle audio problems and support dynamic recognition threshold configuration

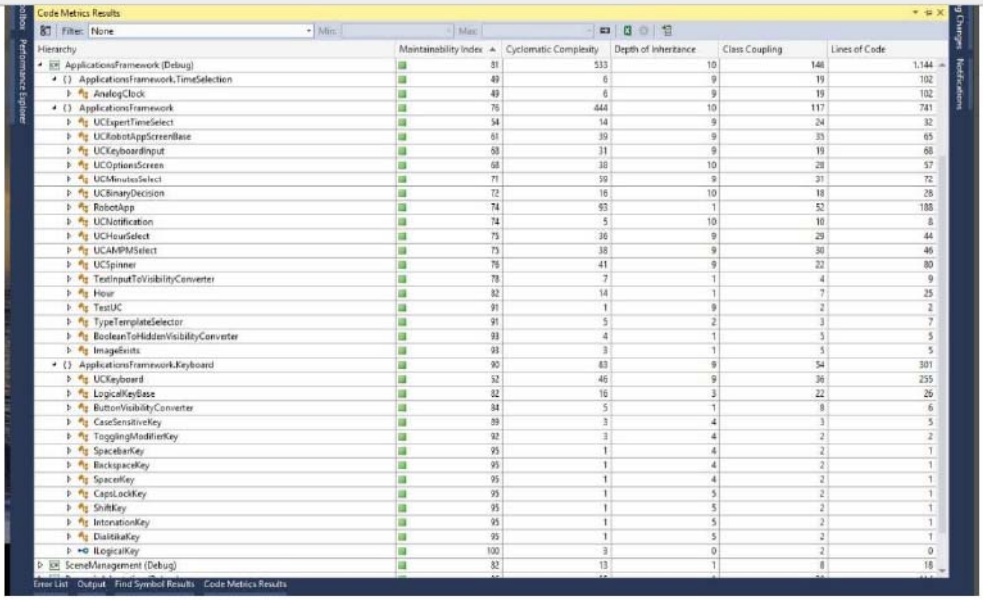
- Develop a gestures recognition modality which should be able to select at runtime the set of supported gestures that can be recognized. Furthermore, since the gesture recognition engine runs on the backend of the robot, all the functionality for creating a bridge between the ROS operating system (which hosts the gesture recognition engine) and the FIRMA framework would have to be developed

- Integrate a rule engine capable of loading and unloading rules at runtime, support manual activation, manual chaining behavior, modality support for the touch, speech and gesture modalities, state management and support for validating rulesets against specific model instances of the respective applications
- Develop a main hosting window for the created application, with support for shortcuts to specific applications or emergency scenarios (e.g. the main menu application, realized as a home button which is always visible, or the emergency call application, realized as an always visible push button) as well as visual cues for the state of the various supported modalities
- Implement the mechanism to support the orchestration of the different application dialogues and screens
- Develop the necessary mechanisms to support adaptive tasks and adaptive component hierarchies
- Develop an adaptation paradigm in the form of dynamic style loading and implement at least one style from all three major style categories (see 5.5.2.3.3), namely the styles base category that is responsible for the overall visual appearance of all the framework elements (visual tree) ~ 355 lines, a coloring scheme style, ~25 lines and a sizing scheme style ~35 lines.
- Implement the integration with FAmINE (see famine code metrics - ~100 lines)
- Develop a virtual on-screen keyboard (~1150 lines)
- Implement an adaptation manager to support the automatic UI adaptation (~200 lines).

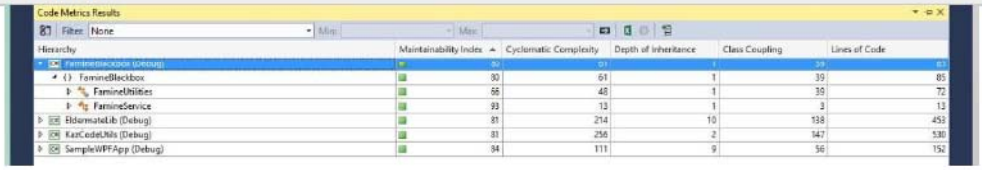
The following table presents some indicative examples of using the Visual Studio embedded functionality for calculating code metrics per project. Based on this

functionality the appropriate calculations where conducted in the case of the alarm clock application.

UI controls



Famine Black Box



Ros Integration

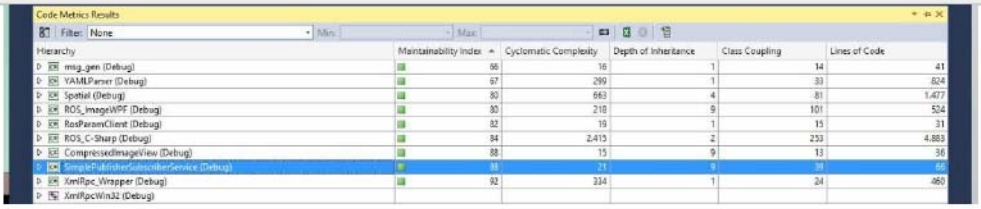


Figure 54: Code calculation metrics in Visual Studio

The following table presents the calculation of the code metrics in the case of a developer that is scripting the applications and its components from scratch.

Component	LOC
Speech synthesis integration	~200
Speech recognition modality and recognition scenarios	~300
Gestures recognition integration (using a predefined gestures set)	~200
Main hosting window	~200

Orchestrator of application screens	~200
Adaptive tasks	~1200
Dynamic style loading and cascading stylesheets	~500
FAmINE integration	~100
UI controls	~800
Virtual keyboard	~400
Adaptation manager	~200
Alarm clock application UI	~500
Alarm clock application logic	~400
Overall development effort	**~5200**

Table 5: Code calculation metrics for developing the alarm clock application from scratch

6.2.2 Developing with the FIRMA framework

Conducted the development of the alarm clock application using the FIRMA framework a number of very critical tasks identified in the previous sections are radically reduced or even eliminated, but some important steps are also added to the development workflow especially for separating the application logic and rules from the application development. More specifically, the tasks for developing the alarm clock application using the FIRMA framework are:

- Use the different interaction modalities writing the code needed to import and parameterize each modality
    - Speech synthesis modality usage
    - Speech recognition modality usage and SRGS grammars development
    - Gestures recognition usage and gestures parameterization
    - ROS interconnection
- Use the main UI navigator window for hosting the created application
- Use the screen orchestrator to manage the displaying of application dialogues on the screen
- Use the UI toolkit to develop the application screens and dialogues for the alarm clock application
- Use the ACTA framework to script the application logic

Component	LOC
Speech synthesis usage	~20
Speech recognition usage	~50
Gestures recognition usage and gestures parameterization	~50
ROS interconnection	~20
Use the main hosting window for the created application	~30
Use the screen orchestrator	~30
Use the UI toolkit to develop the application screens for the alarm clock application	~300
Use the ACTA framework to script application logic	~200
Overall development effort	**~700**

Table 6: Code calculation metrics for developing the alarm clock application using the FIRMA framework

Taking into account the overall development effort calculation as presented in the previous table, the usage of the FIRMA framework could result into an overall reduction of the development effort around ~86% under optimal conditions.

6.2.3 Code assessment metrics & Results

This section reported the results of a simple experiment that was carried out into two distinct phases. Initially an application was developed from scratch taking into account all the required functionality so as to achieve appropriate integration with the robotic platform, integrate several input modalities, be capable of producing adaptive behavior through alternative UI representation and dynamic styling. Then the same application was developed using the FIRMA framework. Using code metrics, the overall development effort calculated by the lines of code a developer should write to achieve a specific result was calculated for both cases. The experiment has proven that the FIRMA framework can achieve up to 86% reduction under optimal conditions. This is achieved in the case of small scale projects where the size of developed product is smaller than the size of the FIRMA framework. In the case of larger scale projects the reduction is still large but in smaller percentages. Furthermore the maximum reduction is achieved when no additions are required to the core framework. In the opposite case developers should also write code to include new features not

supported by the framework. The majority of these situations will fall under the case where new UI components will have to be integrated into the system. In that case the UI components should be developed together with the appropriate style hierarchies so as to support the different adaptation strategies.

6.3 Heuristic Evaluation

Heuristic evaluation - one of the most popular usability inspection methods - is a usability engineering method for identifying usability problems in a user interface design. Heuristic evaluation involves an inspection from a small group of expert evaluators who examine the interface and judge its compliance against recognized usability principles (the heuristics). In general, heuristic evaluation is difficult for a single individual to carry out, because one person will never be able to find all the usability problems in an interface. Experience has shown that different people find different usability problems. More specifically, it has been proved [146] that the sets of usability problems found by different evaluators are in large part non-overlapping. Some usability problems are so easy to find that they can be detected by almost everybody, but there are also some problems that are found by very few and experienced evaluators. Furthermore, one cannot just identify the best evaluator and rely solely on that person's findings, since it is not necessary that the same person will be the best evaluator every time, while some serious usability problems are found by evaluators who may not find other common usability problems. Therefore, it is necessary to involve many evaluators in a heuristic evaluation in order to significantly improve the effectiveness of the method. The ideal number of evaluators that should be used in a heuristic evaluation has been calculated by Nielsen and Landauer [146], who presented a model based on the following prediction formula for the number of usability problems found in a heuristic evaluation:

$$\boldsymbol{ProblemsFound(i) = N * (1 - (1 - l) * i)}$$

Where **ProblemsFound(i)** indicates the number of different usability problems found by aggregating reports from **i** independent evaluators, N indicates the total number of usability problems in the interface, and l indicates the proportion of all usability problems found by a single evaluator.

This formula clearly shows that there is a good payoff from involving more than one evaluator. It is recommended to use three to five evaluators in the procedure, since fewer evaluators cannot detect an adequate number of problems, while using a larger number of evaluators does not provide additional information concerning the usability of the system. The evaluation will be performed by allowing each individual evaluator to inspect the interface alone. After the completion of all evaluation sessions, the evaluators will communicate and aggregate their findings. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator. The results of the evaluation can be recorded either as written reports from each evaluator or by having the evaluators verbalize their comments to an observer as they go through the interface. During the evaluation session, the evaluator examines the interface several times, inspects the various dialogue elements and compares them with a list of recognized usability principles (the heuristics). These heuristics, which are presented below in more details, are general rules that describe common properties of usable interfaces. In addition to the checklist of general heuristics, the evaluators can also consider any additional usability principles or results stemming from their expertise. Furthermore, it is possible to develop category-specific heuristics that apply to a specific class of products as a supplement to the general heuristics. The output of the heuristic evaluation method is a list of usability problems related to the system's user interface, with references to those usability principles that were violated by the design according to the opinion of the evaluators. It is not sufficient for evaluators to simply say that they do not like something; they should explain why they do not like it with reference to the heuristics or to other usability principles.

Many advantages are claimed for heuristic evaluation. One of the most important is that the method provides quick and relatively inexpensive feedback to designers, while the results generate good ideas for improving the user interface. On the other

hand, the development team will also receive a good estimate of how much the user interface can be improved.

Additionally, there is a general acceptance that the design feedback provided by the method is valid and useful. Heuristic evaluation can lead to many usability improvements to take place before a release deadline that would not permit an extensible usability testing involving end users. If the development team is open to new ideas, heuristic evaluation can be an excellent investment of usability resources. One of its great advantages is that it can be performed early on the design process since earlier discovery of usability problems during the development lifecycle helps in identifying and correcting obvious usability problems with very limited costs. Furthermore, carrying out a heuristic evaluation on early prototypes, before actual users are brought in to help with further testing, provides a focus for later usability testing and decreases users' time and effort spent in the usability test.

6.3.1 Rating usability problems

Usability problems' rating according to their importance is a fundamental phase of the heuristic evaluation procedure. As a result, redesign resources will be allocated so as to eliminate the most severe problems first. A problem's severity may be described as a combination of three issues

a) The frequency of its appearance (i.e., frequent vs. rare)
b) The impact of its occurrence (i.e., the difficulty encountered by the user in overcoming the problem)
c) Its persistence (i.e., whether it is a problem that users may overcome if the run into it once, or if they will be repeatedly bothered by it).

In order to determine the severity of usability problems, the following scale is used:

- **0**: It is not a usability problem.
- **1**: It is an aesthetic problem only: unless there is available time, redesigning is not necessary.
- **2**: It is a minor usability problem: eliminating such problems will be assigned a low priority.

- **3**: It is a major usability problem: removing such problems should be given a high priority.
- **4**: It is a usability catastrophe: such problems should be definitely eliminated.

6.3.2 Heuristic rules

The ten general heuristic rules that have been used for the evaluation of this research work are the following [147]:

1. **Visibility of system status:** The system should always keep users informed about on-going procedures, through appropriate feedback and within reasonable time from the relevant actions of the user.
2. **Match between system and the real world:** The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than using system-oriented terms.
3. **User control and freedom:** Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state, without having to go through an extended dialogue.
4. **Consistency and standards:** Users should not have to wonder whether different system messages, situations, or actions mean the same thing. Thus, their interpretation should be unique and intuitive for the users.
5. **Error prevention:** A careful design, which prevents a problem from occurring in the first place, is even better than helpful error messages.
6. **Recognition rather than recall:** Objects, actions and options should be visible. The user should not have to remember information from one part of the interface dialogue to another. Instructions for use should be visible as well, or easily retrievable when appropriate.
7. **Flexibility and efficiency of use:** The system should cater to both inexperienced and experienced users.
8. **Aesthetic and minimalist design:** Information that is irrelevant or rarely needed should not be presented at first level.

9. **Help users recognize, diagnose and recover from errors:** Error messages should be expressed in plain language (no codes), precisely indicating the problem, and constructively suggesting a solution.
10. **Help and documentation:** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

6.3.3 Methodology followed

In order to evaluate the UI controls of the FIRMA framework the following procedure was followed:

- A UI window was developed in order to host a demonstrator application for the evaluators
- From the UI window a list of buttons became available to the evaluators so as to select the UI control to evaluate
- By pressing one of the buttons a new window was opened displaying on the center the control to be evaluated
- A drop down menu was available to the evaluators that contained a number of pre-defined profiles. The selection of an option from the drop down menu resulted to the adaptation of the user control to the selected profile

This process was preferred mainly because it made easier for the evaluators to mark the identified usability errors by just filling in a table the name of the control, the selected profile and the error.

For performing the evaluation three usability experts used the presented application and through the application inspected all the available controls and recorded the identified usability problems. These problems were gathered per control and graded based on their severity.

6.3.4 Evaluation results

6.3.4.1 AM/PM specifier

The following table presents the usability errors identified during the evaluation of the "AM/PM specifier" user control.

Issue	Severity score	Profile	Suggestion
Although the contrast used to mark the selected option is adequate, the contrast for the not selected one is not adequate.	1	All profiles	Make sure that the contrast is appropriate in both cases and that the user understands which contrast represents the selected and which the not selected option
The AM/PM options are grouped with a container that has a border and background. The text is outside the container. This in some cases may confuse the user.	1	All profiles	It will be more appropriate to encapsulate everything within the border so to mark the boundaries of the control and give better visual information to the user.
Buttons seem stretched within the border of the control.	1	All profiles	Consider increasing the

Furthermore the distance between the buttons seems limited			default size of the control and adding extra padding around and between the controls. This will also enhance the usage of the control on touch devices
The default colour of the not selected button (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour

Table 7: AM/PM specifier heuristic evaluation results

6.3.4.2 *The Binary decision dialogue*

The following table presents the usability errors identified during the evaluation of the "Binary decision dialogue" user control.

Issue	Severity score	Profile	Suggestion
There is an issue with the colour contrast of control buttons especially in the case where textual descriptions are inserted. The white on grey produces poor contrast.	2	All profiles	Use a tool to identify the appropriate colour contrasts per profile and remain consistent on these selection for all the controls.

On the demo application text seemed stretched towards the bottom side of the button while having enough threshold on the top side.	1	All profiles	Just increase the threshold on the bottom side to match the one of the top side of the button.
The distance between the buttons seems limited	1	All profiles	Consider adding extra padding around and between the controls. This will also enhance the usage of the control on touch devices
The default colour of the buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour

Table 8: The Binary decision dialogue heuristic evaluation results

6.3.4.3 Calendar

The following table presents the usability errors identified during the evaluation of the "Calendar" user control.

Issue	Severity score	Profile	Suggestion
Calendar in not always readable	3	Low ambient light	Consider increasing font size in the

			appropriate profiles
Calendar text is not scaled consistently. Although not evaluated in the same window it seems like the calendar text is not scaling in the same percentage as occurring in other controls.	3	All profiles	Make changes that happen to the calendar control consistent with the changes that happen to other user controls.
Embedded calendar buttons are difficult to use. Calendar buttons will be hard to use on a touch screen even for users with no functional limitations because they are too small.	4	All profiles	Embedded buttons should at least triple in size.
The distance between the day links seems limited especially when touch devices are used.	1	All profiles	Consider adding extra padding around and between the day links. This will also enhance the usage of the control on touch devices

Table 9: Calendar heuristic evaluation results

6.3.4.4 *Analogue clock*

The following table presents the usability errors identified during the evaluation of the "Analogue clock" user control.

Issue	Severity score	Profile	Suggestion
Time is not always easy to perceive	3	All profiles	Consider adding numbers to indicate time and make clock hands thicker. Furthermore eliminate subdivisions between every five minutes
In the case of high levels of ambient lighting, the hands of the clock might not be visible	3	Ample ambient lighting profile	Consider eliminating color coding and using black for clock hands

Table 10: Analogue clock heuristic evaluation results

6.3.4.5 *Expert time selection dialogue*

The following table presents the usability errors identified during the evaluation of the "Expert time selection dialogue" user control.

Issue	Severity score	Profile	Suggestion
The drop down menu that appears on the right side of the control seems to have different text size than the one used to indicate selected time	1	All profiles	Keep text sizes consistent
The drop down on the left side is not perceived as one until	3	All profiles	Consider altering the control and

someone touches it. This is mainly due to the very small size of the down arrow			increasing the arrow size according to the selected profile
Buttons plus and minus should have a more three dimensional look so as to be more easily perceived as a place where interaction should happen	2	All profiles	Consider increasing shadow depth
Borders that distinguish hours from minutes are not always visible	2	Low ambient light	Consider making borders thicker or painting borders with a more intense value of grey
The default colour of the buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour

Table 11: Expert time selection dialogue heuristic evaluation results

6.3.4.6 *Virtual keyboard User Control*

The following table presents the usability errors identified during the evaluation of the "Virtual Keyboard" user control.

Issue	Severity score	Profile	Suggestion
Text is not readable in most of the profiles (especially considering usage by elder users)	3	All profiles	Consider increasing text size in all profiles

Buttons should be made more visible	2	All profiles	Increase the contrast of button borders
The QWERTY layout is not consistent in all profiles	2	Low ambient light	Keep layout consistent between profile changes
Not all symbols are available	1	All profiles	Consider providing another view to access the full set of symbols
The default colour of the buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour

Table 12: Virtual keyboard user control heuristic evaluation results

6.3.4.7 *Virtual keyboard input dialogue*

For this control the issues that were identified for the Virtual keyboard remain the same. This section identifies the issues of the dialogue that encapsulates the keyboard so as to become an input dialogue.

Issue	Severity score	Profile	Suggestion
The user may not understand that the keyboard is popped-up as a dialogue	1	Low ambient light	Consider providing enough contrast so as the dialogue remains always visible
The contrast of the text field is very limited. The “Enter your	2	All profiles	Increase contrast

text here" message is not always visible			
The text field seems two dimensional. It is not certain that users will perceive it as a place to enter text	2	All profiles	Insert an internal shadow to the text field so as to look like a placeholder for text

Table 13: Virtual Keyboard input dialogue heuristic evaluation results

6.3.4.8 Auto dismissible notification

The following table presents the usability errors identified during the evaluation of the "Auto dismissible notification" user control

Issue	Severity score	Profile	Suggestion
The text size and contrast may not be adequate for usage together with some user profiles	2	Especially in the case of low ambient light	Increase contrast and possibly consider increasing text size in some profiles

Table 14: Auto dismissible notification heuristic evaluation results

6.3.4.9 Hour selector

The following table presents the usability errors identified during the evaluation of the "Hour selector" user control

Issue	Severity score	Profile	Suggestion
Although the contrast used to mark the selected hour is	1	All profiles	Make sure that the contrast is appropriate in all

adequate the contrast for the not selected is not			cases and that the user understands which contrast represents the selected and which the not selected hour
The external padding is not the same with the internal padding	1	All profiles	Aesthetically making these distances the same will improve the look and feel of the control
The default colour of the not selected buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour

Table 15: Hour selector heuristic evaluation results

6.3.4.10 Minutes selector

The following table presents the usability errors identified during the evaluation of the "Minutes selector" user control

Issue	Severity score	Profile	Suggestion
Although the contrast used to mark the selected minutes is adequate the contrast for the not selected is not.	1	All profiles	Make sure that the contrast is appropriate in all cases and that the user understands which contrast

			represents the selected and which the not selected minutes
The external padding is not the same with the internal padding	1	All profiles	Aesthetically making these distances the same will improve the look and feel of the control
The default colour of the not selected buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour
The units selection dialogue may cause issues in selecting the desired value	2		Consider evaluating this control with real users
The dashes in the dozens selection dialogue are completely unintuitive and can be easily confused for the sign of subtraction	4	All profiles	Consider removing the dashes and add a zero for the units as if the final selected value comes from adding the dozens value with the units value.

Table 16: Minutes selector heuristic evaluation results

6.3.4.11 Options presenter

The following table presents the usability errors identified during the evaluation of the "Options presenter" user control

Issue	Severity score	Profile	Suggestion
The options are not clearly identified as buttons	2	All profiles	Consider adding a more three dimensional look and feel to the buttons so as to be clearly distinguishable
Each option seems to have a separate background behind each button. This introduces fuzziness to the interface	2	All profiles	Consider eliminating the background
The next and back options cover a large area of the display. This might make interfaces with a lot options harder to use (a lot of pages must be generated so as to host all the options)	2	All profiles	Consider providing a developer controlled adaptation where the next and back buttons are located under the options so as to have space for twelve concurrently presented options

The default colour of both the options and the next/back buttons might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour
In some case the control may be used to present items that are not named as options in the context of a specific interface	2	All profiles	Consider providing an option for the developer to change message text

Table 17: Options presenter heuristic evaluation results

6.3.4.12 Spinner

The following table presents the usability errors identified during the evaluation of the "Spinner" user control

Issue	Severity score	Profile	Suggestion
The "plus" and "minus" options are not clearly identified as buttons	2	All profiles	Consider adding a more three dimensional look and feel to the buttons so as to be clearly distinguishable
The caption of the spinner is located on the bottom side of the control. In some cases developers may wish it to be positioned elsewhere	1	All profiles	Consider adding a caption placement option to the control

In some profiles (e.g. novice user) the plus and minus options may be confusing	2	Novice user	Consider altering symbols with text (e.g. next-previous)
The default colour of the buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour

Table 18: Spinner heuristic evaluation results

6.3.4.13 Tile button

The following table presents the usability errors identified during the evaluation of the "Tile button" user control

Issue	Severity score	Profile	Suggestion
Tile button is not clearly identified as button	2	All profiles	Consider adding a more three dimensional look and feel to the buttons
Tile button presents a border around the image that changes between profiles. In some cases the developer might need access to this functionality	1	All profiles	Consider allowing the developer to override just the border colour value for each profile
Tile button has curves on each angle. In some cases the developer might need square	1	All profiles	Consider allowing the developer to select the desired corner radius

buttons for his application but with the same functionality			

Table 19: Tile button heuristic evaluation results

6.3.4.14 Time selection dialogues

The following table presents the usability errors identified during the evaluation of the "Time selection" user control

Issue	Severity score	Profile	Suggestion
The external padding is not the same with the internal padding	1	All profiles	Aesthetically making these distances the same will improve the look and feel of the control
The default colour of the buttons (grey) might lead to misinterpretation (the user may understand that the button is disabled)	2	No adaptation	Consider altering the default button colour
The layout used in all profiles is horizontal (first hour then minutes and then AM or PM). This may result into space or layout limitations for some application.	2	All profiles	Consider allowing the developer to change the placement or just the orientation of the time selector user control

Table 20: Time selection dialogues heuristic evaluation results

6.3.4.15 UI navigator

The following table presents the usability errors identified during the evaluation of the "UI navigator" user control

Issue	Severity score	Profile	Suggestion
The options that appear on the bottom side of the container are not clearly identified as buttons	2	All profiles	Consider adding a more three dimensional look and feel to the buttons so as to be clearly distinguishable
The layout used in all profiles is horizontal. This layout option may not be always what the designer or the developer wishes for his/her application	2	All profiles	Consider providing alternative navigation templates and allowing the developer to select the most appropriate for his application

Table 21: UI Navigator heuristic evaluation results

6.3.5 Discussion

This section has presented the process followed and outcomes of the heuristic evaluation of the FIRMA framework. The results of the evaluation were rather positive in terms of the overall acceptance of the framework by the evaluators and all the identified usability errors were clearly defined and documented. Based on the feedback received a second iteration to the development of the UI framework was conducted in order to fine tune the controls. This preliminary evaluation should be considered as an intermediate step to the overall process of evaluating the outcomes of this research work. The final validation will occur by a larger scale user based evaluation.

The final validation will take place in the context of the European RAMCIP project trials that will take place during a six month period in Spain (Barcelona, Fundacio ACE, Barcelona Alzheimer treatment and research center) and in Poland (Lublin Medical University). The end users will be healthy elderly volunteers and MCI/early AD patients that will interact with the RAMCIP robotic platform in controlled environments during the time period April 2017 – October 2017. The UI of the RAMCIP platform will be based on the FIRMA framework and will be evaluated and validated in terms of ease of learning and ease of use, comfortable perception, acceptability and satisfaction.

7 Case-study applications based on the FIRMA framework

This chapter presents the outcomes of this research word in the context of two implemented applications that were based on the FIRMA framework. To this end, this chapter presents the way that the implemented by this research work infrastructure and framework tools and elements are used to facilitate the needs of the creation of two case study applications, namely the alarm clock application and the TV control application. The first application is the alarm clock application that can be used for managing a user's daily tasks scheduled for specific times of the day. The second application is the TV control application that can be used for managing the television sets around the house. The TV control application uses a FAmINE service that exposes the necessary functionality regarding the control of the different TV sets.

The test case applications have been hosted into differently styled UI Navigator windows according to the adaptive style hierarchies approach that is discussed in the previous chapter.

Figure 55: The Main Menu hosted into differently styled UI Navigator windows

Figure 55 shows the two main UI Navigator windows hosting the two test case applications. The windows have been styled differently to demonstrate the styling capabilities of the framework.

7.1 The Alarm Clock Application

7.1.1 Preliminary low fidelity paper based prototyping

The alarm clock application is responsible for displaying the current time as well as managing the daily alarms of the user in respect to the time of the day. The user is able to see the current time, see his daily alarms, add new ones, delete existing alarms and snooze elapsed alarms. The application supports all available interaction modalities and is adaptable and adaptive to fit the needs of the users.

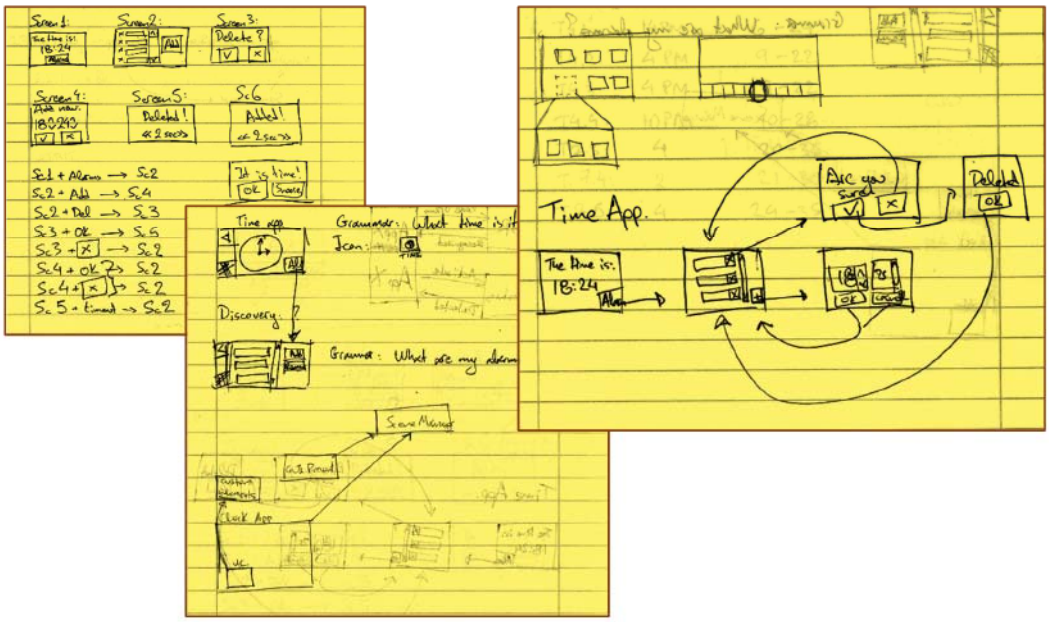


Figure 56: Preliminary low fidelity paper based prototyping for the Alarm clock Application

Figure 56 shows the mockups from the low fidelity paper based prototyping approach regarding the design, and the states of the internal FSM of the Alarm clock application that led to the implementation of the final application.

7.1.2 Implementation

The selected designs that stemmed from the low fidelity paper based prototyping regarding the Alarm Clock Application, drove the implementation of the application.

The adaptations that have been implemented for this test case application concern the coloring scheme of the application which changes depending on the time of day, on the level of the ambient lighting in the surrounding environment and on whether the user is wearing his/her glasses or not. During the day, or when the lighting level of the room increases (the lights are turned on), the coloring scheme of the application changes to a lighter scheme that is easier for the eyes in order for the user to be able to see more clearly. Furthermore, when the level of ambient lighting is reduced, the application automatically changes into a darker color scheme with higher contrast to compensate for the lighting changes.

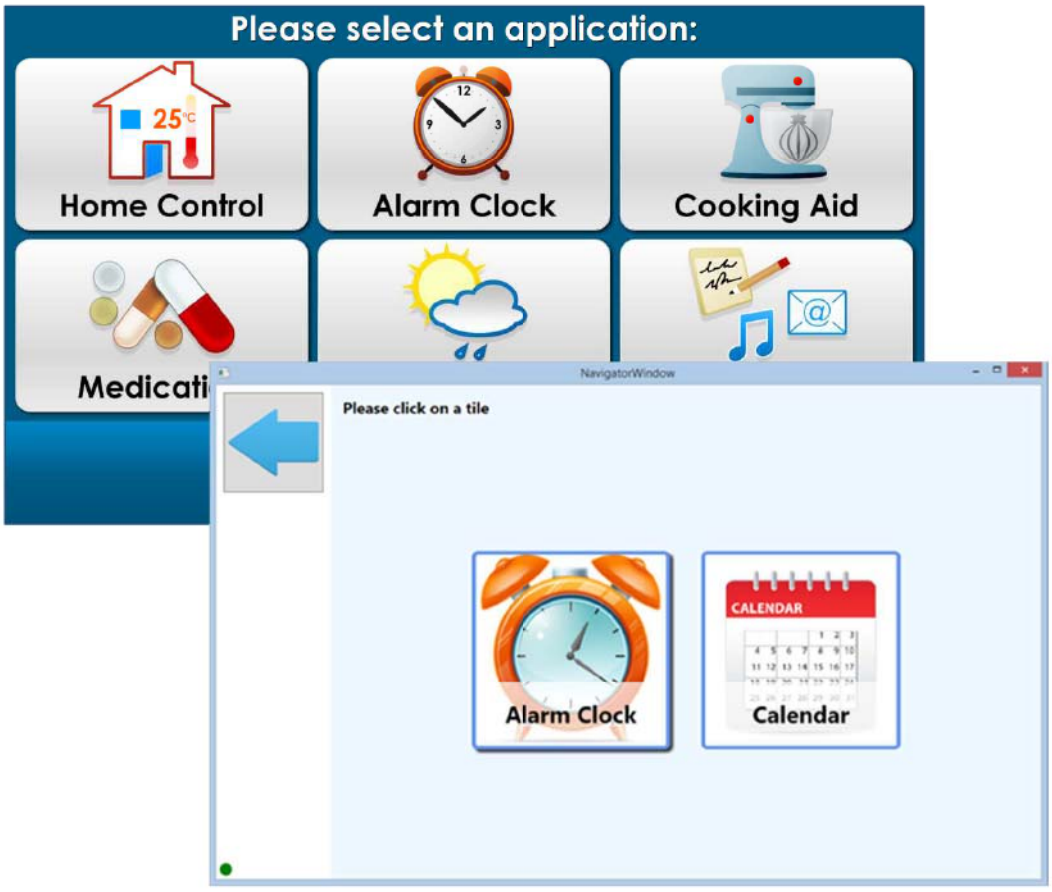


Figure 57: The main menu showing the alarm clock application (top), the "schedule management" submenu displaying the alarm clock icon as an alternative to hosting all the applications in the main menu (bottom)

Finally, when the robot detects that the user is not wearing his/her prescription glasses, the coloring scheme changes to a high contrast scheme to facilitate the user. Similarly, the size of the used controls, dialogues and messages change in respect to the respective pose of the user and the distance between the user and the robot to allow for comfortable interaction.

Figure 58: The different coloring and sizing schemes supported by the alarm clock application

When the user is sited, the size of the used controls, dialogues and messages adapt according to the distance between the user and the robotic platform. The application supports three different sizes, a large sized scheme for bigger distances, a medium sized scheme for average distances and a small sized scheme for a more comfortable interaction when the robotic platform is very close to the user. Furthermore, the

application supports a dark colored scheme for the night and a light colored scheme for the day. Moreover, when the robot detects that the user is not wearing his/her glasses, the application's scheme changes to a high contrast coloring scheme for convenience.

Figure 57 shows the alarm clock application icon both as part of the main menu and as part of the "Schedule Management" category (submenu) of the main menu as opposed to hosting all the available application in one main menu.

Figure 58 shows the different coloring and sizing schemes that the alarm clock application supports.

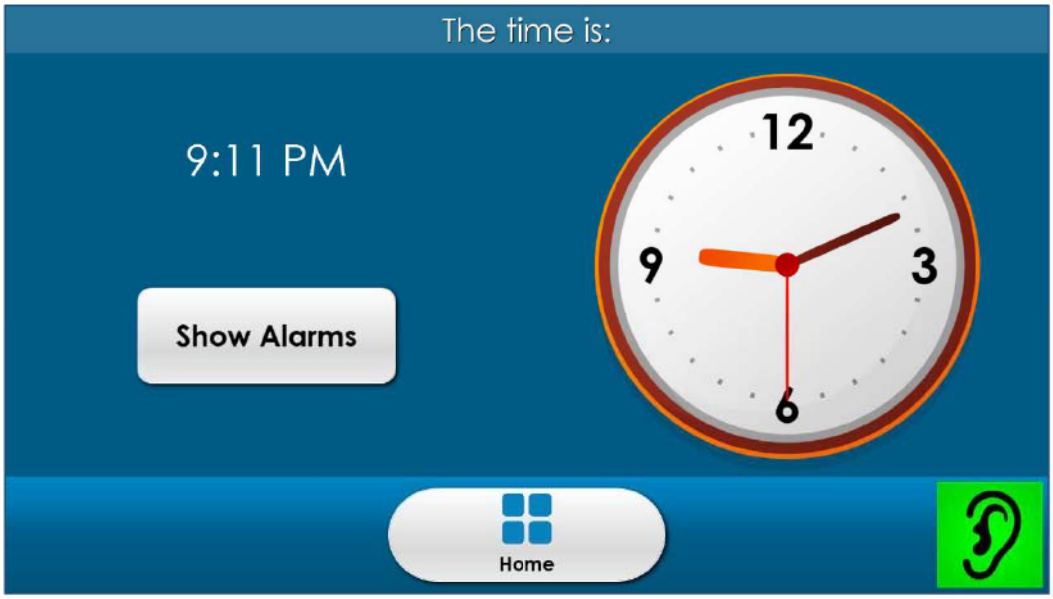


Figure 59: The Alarm Clock Application: Displaying time

Figure 59 shows the clock screen of the alarm clock application. On the lower bottom right corner of the window, a green visual cue representing the status of the speech recognition modality can be seen. The speech recognition modality is active, hence the green "ear" icon is visible. The clock screen displays the current time of the day. The coloring scheme of the application is dark due to the bright ambient lighting and the fonts and buttons that have been displayed are big enough for the user to be able to see while being in a standing position beside the robotic platform. The Home button of the main UI Navigator window is enabled which shows that the user has navigated

away from the home screen of the main menu and the green speech recognition modality cue displays that the modality is active.

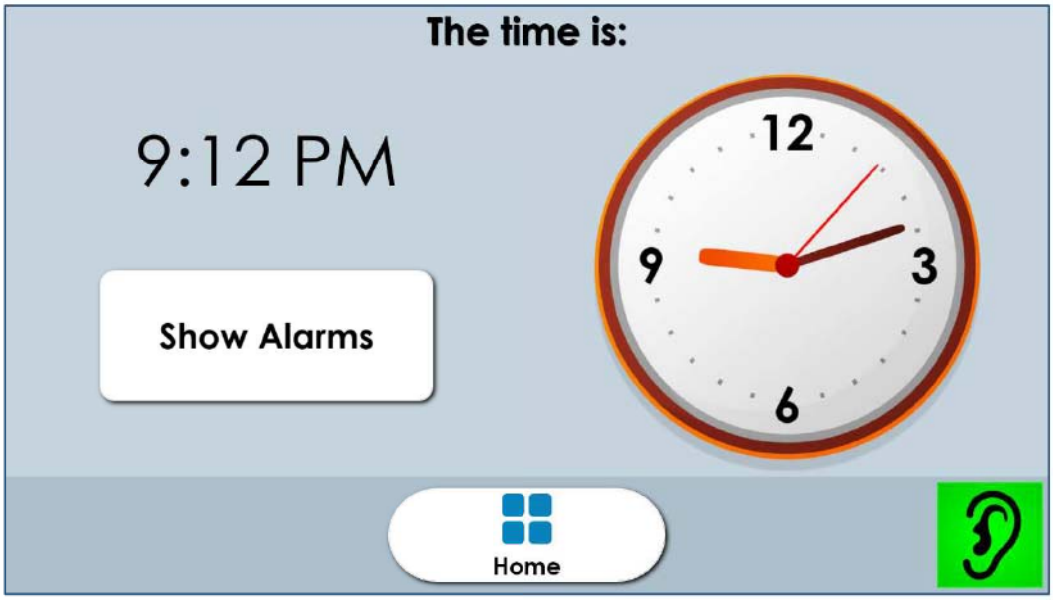


Figure 60: The Alarm Clock Application: Displaying time while adapting to the user position and the ambient lighting

Figure 60 shows the same main clock screen of the alarm clock application, displayed using a different coloring scheme and different sizes for the various framework elements, controls and messages that were used. The coloring scheme has changed to brighter colors in accordance with the adaptation policy in cases of darker ambient lighting conditions, whereas the sizes of the different controls has been reduced to accommodate the fact that the user is sitting in a chair and the robot has navigated in front of him. The home button of the window is active because the user is not in the main menu screen and the speech recognition modality is being displayed as active. The user can press the home button to return to the main menu of the robot or press the "Alarms" button to navigate to the screen that displays the current schedule of the day. Since the speech recognition modality is active, the user can also give verbal commands to the robot.

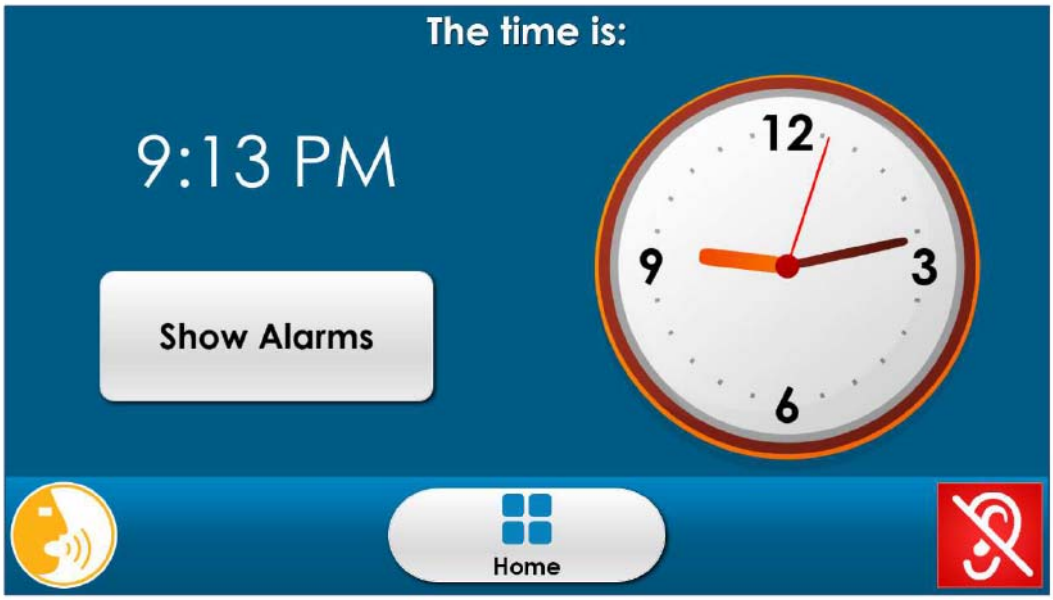


Figure 61: The Alarm Clock Application: Using the speech output modality to verbally tell the time

Figure 61 shows the exact opposite adaptation policy in contrast to Figure 60. The application is using the dark color scheme and the sizes of the various controls have been increased due to the fact that the user is now standing. Furthermore, on the lower bottom left corner of the main window, a visual cue regarding the speech synthesis modality is being shown. The above clock screenshot displays the alarm clock screen when the robot is telling the time to the user using the speech synthesis output modality. The modality is adapted to the context of use and the preferences of the user (volume, rate, pitch, gender of the used voice) and the modality status is depicted by the yellowish round icon which displays the robot talking to the user. As a result, the adaptation manager has temporarily deactivated the speech recognition modality in order to avoid the case where the robot might recognize its own speech as a dictated command. As a result the visual cue that represents the speech recognition modality has been changed into a red disabled icon.

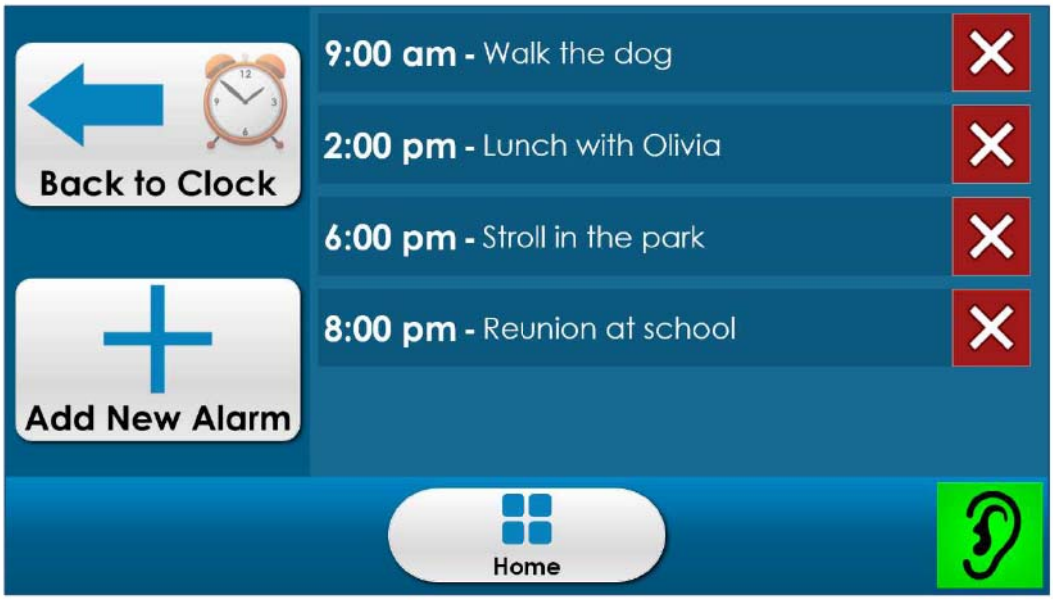


Figure 62: The Alarm Clock Application: Daily alarms overview

Figure 62 shows the alarms screen of the alarm clock application. In the middle, the user can see a list containing all the daily alarms that are active for the respective day. For each alarm, the time of the alarm and an assigned message that describes it is being displayed. The user has the liberty to select a desired alarm and press the red "x" button next to it in order to delete it. Before the deletion of the selected alarm, a confirmation message is displayed. Furthermore, the user can press the "Back to clock" button (containing both an image and a text description) in order to return to the previously displaying screen, the "Add new alarm" button to add a new alarm or the home button to return to the main menu screen. The speech synthesis modality is active and the robot is ready to accept verbal commands from the user, while the speech synthesis modality is inactive as the robot is not currently communicating any auditory information to the user.

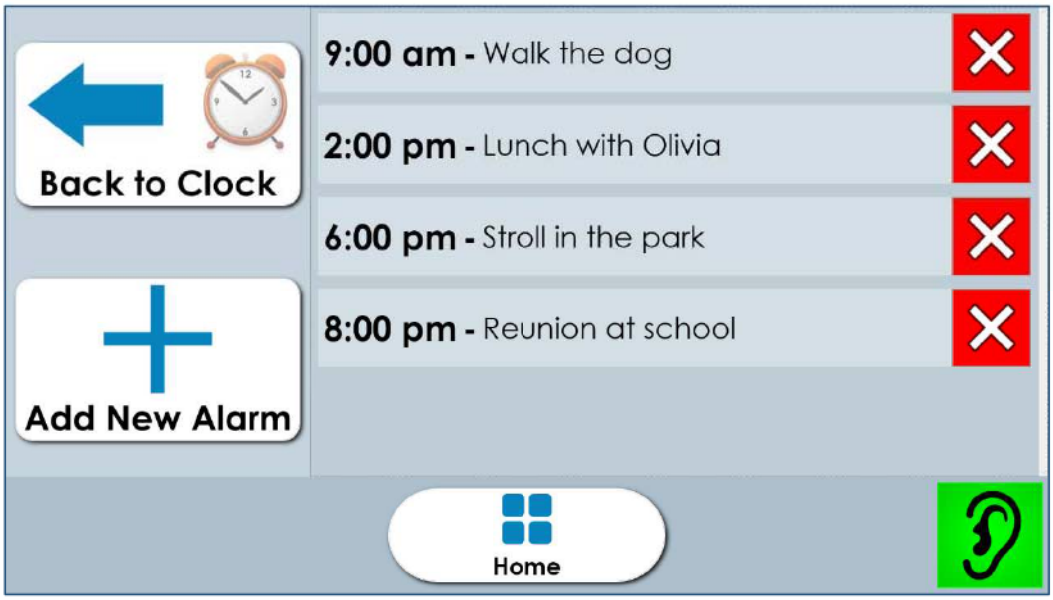


Figure 63: The Alarm Clock Application: Daily alarms overview adapted to the ambient lighting of the room

Figure 63 displays the same daily alarms overview screen as Figure 62 but the coloring scheme of the application has been automatically adapted to the low ambient light conditions of the room at the respective time.

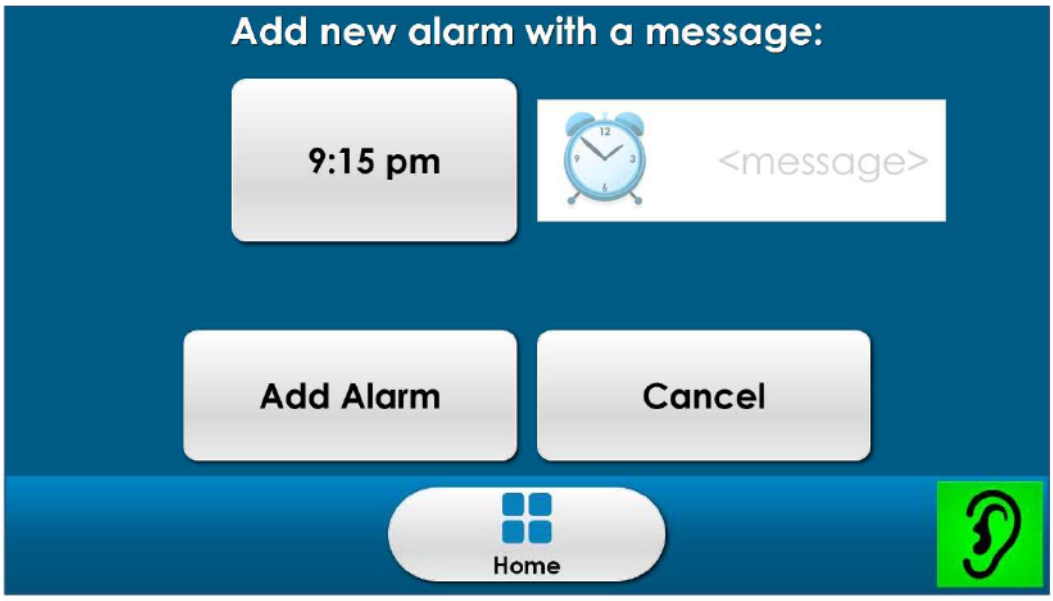


Figure 64: The Alarm Clock Application: Adding a new alarm

Figure 64 shows the alarm adding screen of the alarm clock application. The screen displays the current time of the system to facilitate the user when he wants to add an alarm at a relatively short time span. Having set the alarm time to match the respective time of the day makes it easier for the user to apply small time deltas to create alarms that are differ in small timespans. Furthermore, the user can specify a custom message to assign o the alarm that is being created. The user can specify the desired alarm time by pressing on the button displaying the alarm time. By pressing the alarm time button the time selection process is initiated, where the user is guided step by step throughout the time selection procedure according to his preferences and level of experience. The time selection process is automatically tailored to the end user while the respective adaptive task hierarchy is instantiated step by step.

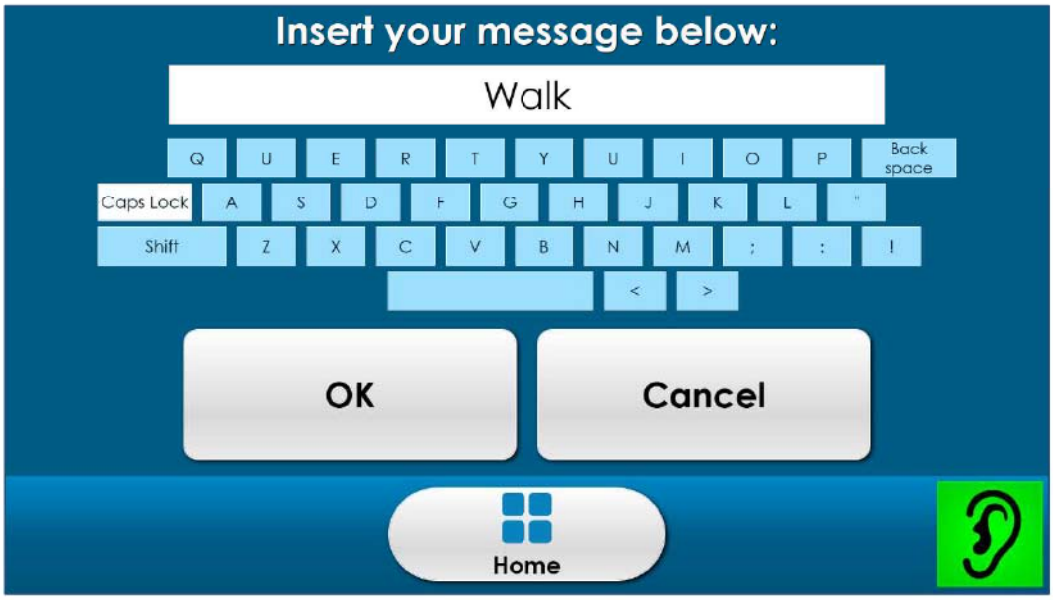


Figure 65: The Alarm Clock Application: Specifying a message to assign to the new alarm

When the user has selected the desired time for the alarm he can press the “Add alarm” button to confirm the creation of the alarm, or the “Cancel” button to return to the previously displayed screen.

The user is given the functionality to add a message to be assigned to the alarm being created by pressing on the message textbox that is displayed on the screen. When the user presses the textbox, the screens changes into the one that is displayed in Figure

65. The screen changes to display a virtual on-screen keyboard that the user can use to input the desired alarm message. When the user finishes entering the message, he can either press the “Accept” button to accept the message and return to the previous screen or the “Cancel” button to return to the previous screen without accepting the changes.

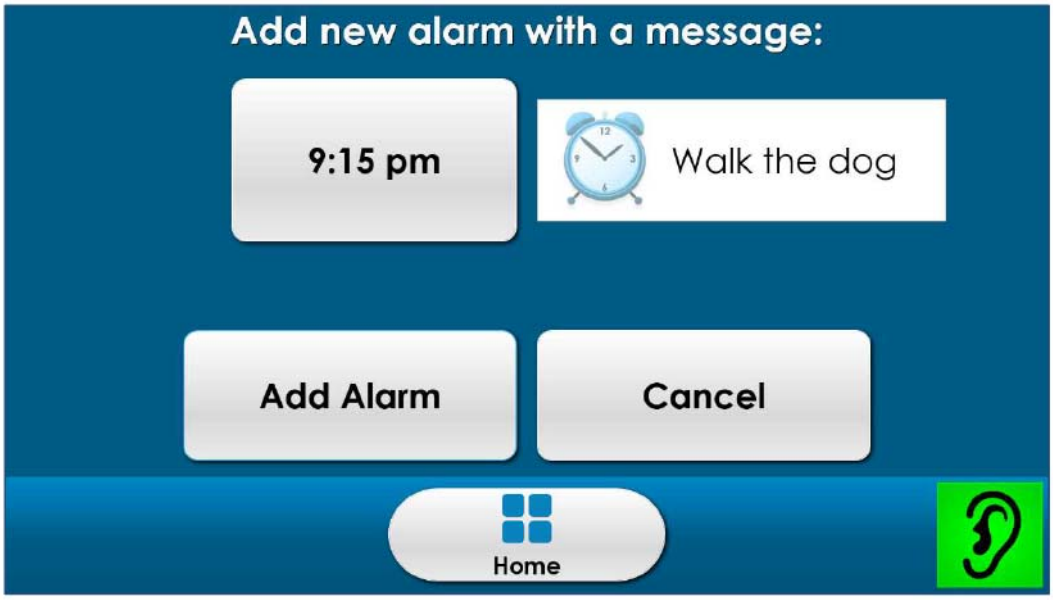


Figure 66: The Alarm Clock Application: A message for the new alarm has been assigned

Figure 66 shows the alarm clock creation screen where the user has accepted the changes regarding the message that is assigned to the alarm being created.

When the user presses the alarm time button, the time selection process is initiated. Figure 67 displays the time selection process for the experienced user. The user is presented with spinners and a drop down box to select and adjust the desired time in terms of hour, minutes and the am/pm time specifier. The time selection process is adapted to the experience of the user. Different controls and additional steps can be automatically selected by the framework for average or inexperienced users.

Figure 67: The Alarm Clock Application: Specifying the time of the new alarm

The user can select the desired time and then accept the changes or reject them to return to the previous dialogue. The user is able to cancel at any time, or press the home button to return to the main menu screen. The speech recognition modality is enable, therefore the speech recognition visual cue is green.

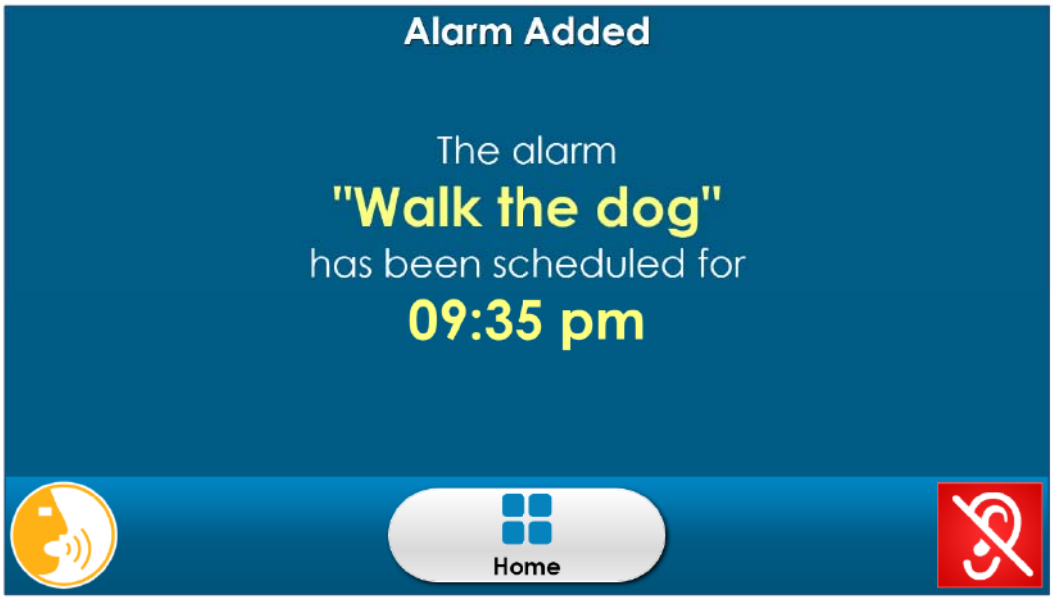


Figure 68: The Alarm Clock Application: Alarm added notification

Figure 68 displays the confirmation dialogue when the user acknowledges the creation of a new alarm. The notification include the time of the alarm as well as the specified assigned alarm message.

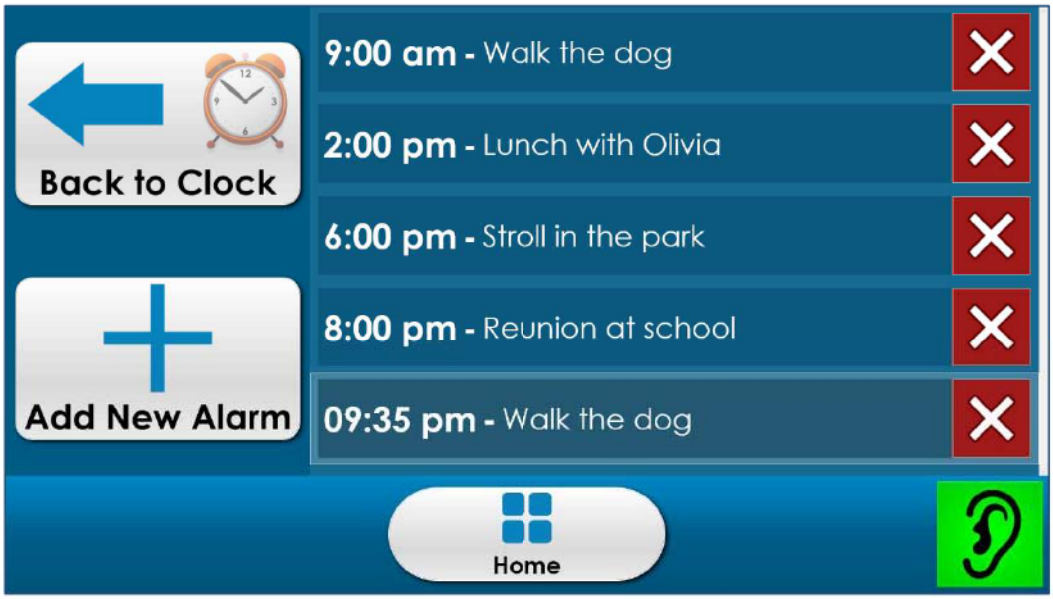


Figure 69: The Alarm Clock Application: Daily alarm overview showing the newly added alarm

Figure 69 displays the daily alarm overview screen showing the newly added alarm. The alarm is selected which is displayed by changing the alarm message foreground and background color.

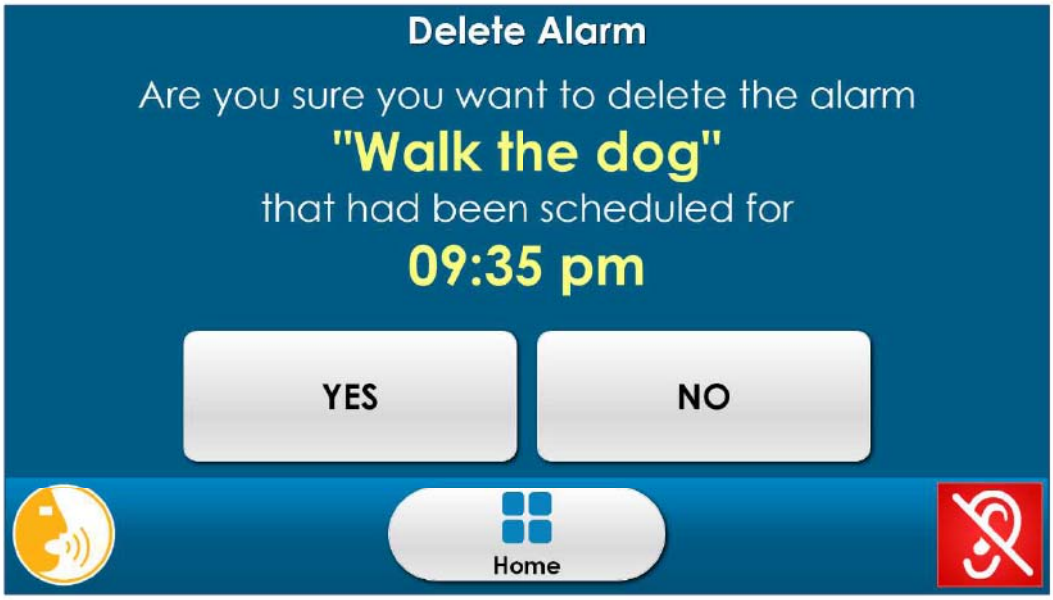


Figure 70: The Alarm Clock Application: Confirmation for deleting an existing alarm

Figure 70 displays the confirmation dialogue when the user is about to delete an existing alarm. The dialogue contains the time and the assigned message of the alarm that is going to be deleted. The user can acknowledge the deletion or press "Cancel" to return to the previous dialogue showing the alarms overview. The robot uses the speech synthesis modality for redundancy purposes to verbally convey the same information to the user. The speech recognition modality stays disabled while the robot is talking to prevent false positive recognitions of the speech recognition engine.

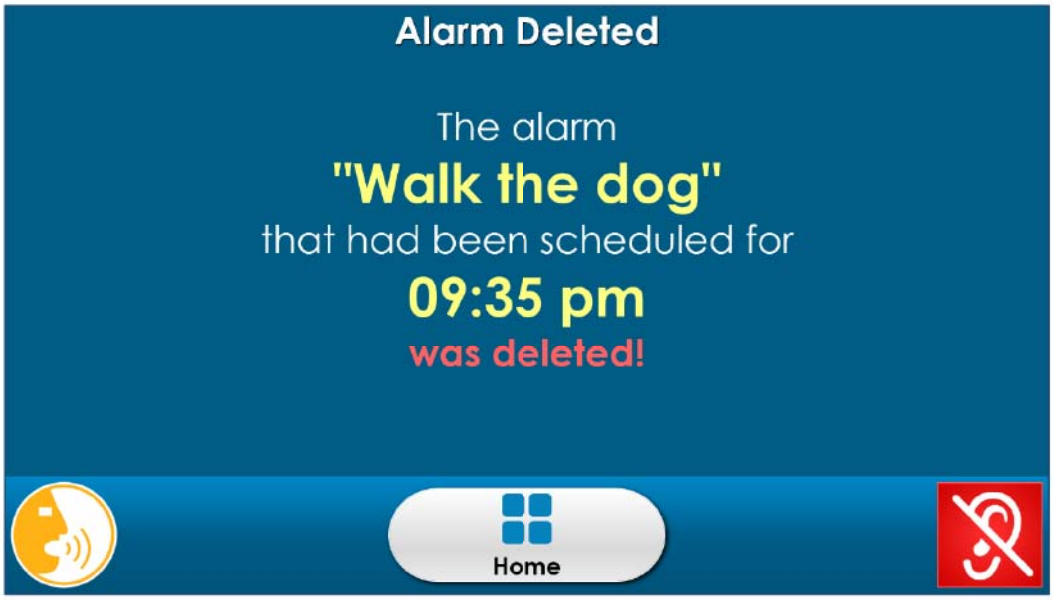


Figure 71: The Alarm Clock Application: Alarm deletion notification

Figure 71 displays the deletion notification when the user deletes an alarm. Similarly to the deletion confirmation dialogue, the deletion notification dialogue contains the time and the message that was assigned to the alarm that has just been deleted. The robot uses to speech synthesis modality to deliver auditory feedback to the user for redundancy purposes while the speech recognition modality remains deactivated throughout the speech synthesis delivery. Finally the deletion notification is automatically dismissed by the robot when the speech interaction is completed.

Figure 72: The Alarm Clock Application: Alarm elapsed notification

Figure 72 displays the elapsed alarm screen. When a specified alarm time matches with the time of the system, the alarm is marked as elapsed and the elapsed alarm confirmation dialogue is displayed to the user. The user has the option to either acknowledge the elapsing of the alarm in which case the alarm is considered as dismissed and it is removed from the list of daily alarms, or chose to snooze the alarm which leads to the next figure.

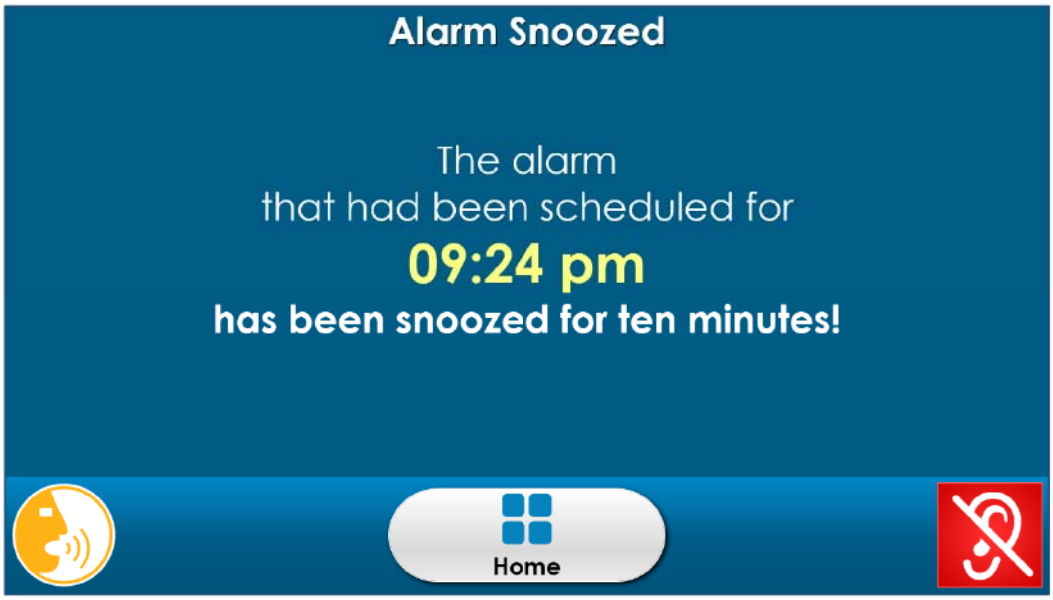


Figure 73: The Alarm Clock Application: Alarm snooze notification

Figure 73 displays the alarm snoozed notification dialogue that informs the user that the elapsed alarm has been snoozed for ten minutes. Similarly to the alarm deleted notification dialogue, the alarm snoozed notification dialogue contains the original time of the alarm and the potential assigned message. The robot uses the speech synthesis modality to provide additional auditory feedback to the user, whereas the notification dialogue is automatically dismissed when the robot finishes talking to the user. The speech recognition modality remains inactive during the whole speech synthesis process.

7.2 The TV Control Application.

7.2.1 Preliminary low fidelity paper based prototyping

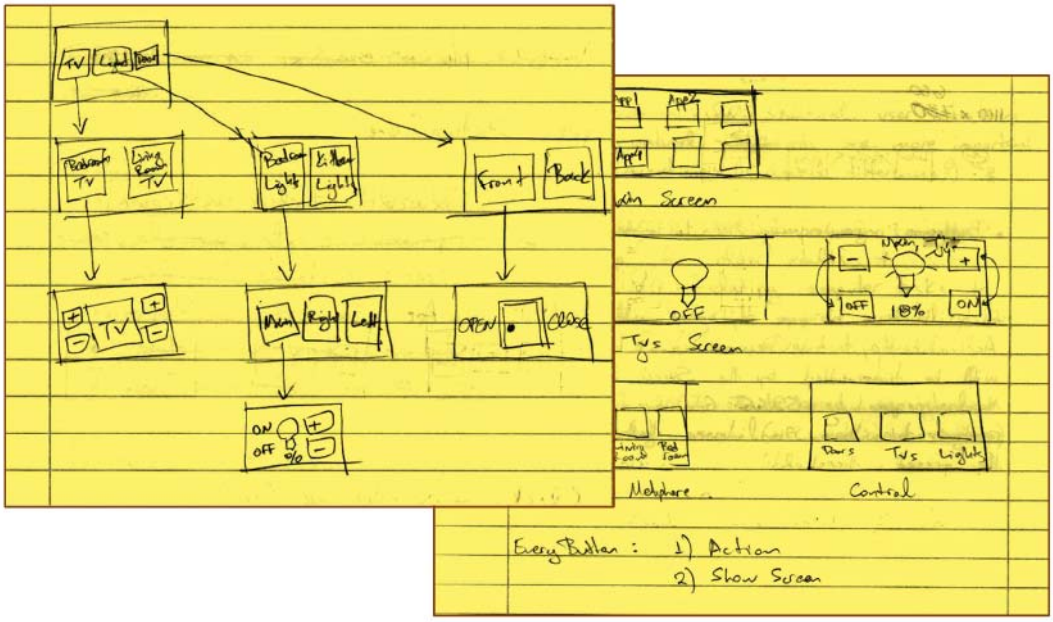


Figure 74: Preliminary low fidelity paper based prototyping regarding the design of the TV control application

The TV control application was developed mainly to showcase the FIRMA framework's capabilities regarding the inherent support of the FAmINE ambient intelligence network environment. In this scope, an application was developed that offers to the users the functionality to remotely control any TV inside the house by exploiting the functionality exposed by an ambient intelligence service running in the environment. Figure 74 shows the preliminary design on paper regarding the TV control application. From this prototyping effort, the final design of the application was selected.

7.2.2 Implementation

After the preliminary paper based low fidelity prototyping regarding the TV control application, the selected designs drove the implementation of the selected functionality. To this end, a TV control service was developed that offers the functionality to remotely control a TV in the context of switching it on and off, controlling the level of the TV's volume as well as controlling and changing the respectively selected TV channel. This functionality was exposed through the means

of a CORBA service in the context of a FAmINE service that the application transparently uses to control the TV.

Figure 75: The main menu hosted inside a differently styled UI Navigator window.

Figure 75 shows the main menu of the robot being hosted into a differently styled UI Navigator window to demonstrate the unlimited styling options that the FIRMA framework has, based on the cascading style sheets approach. Coloring and styling schemes can be easily added and incorporated into the framework.

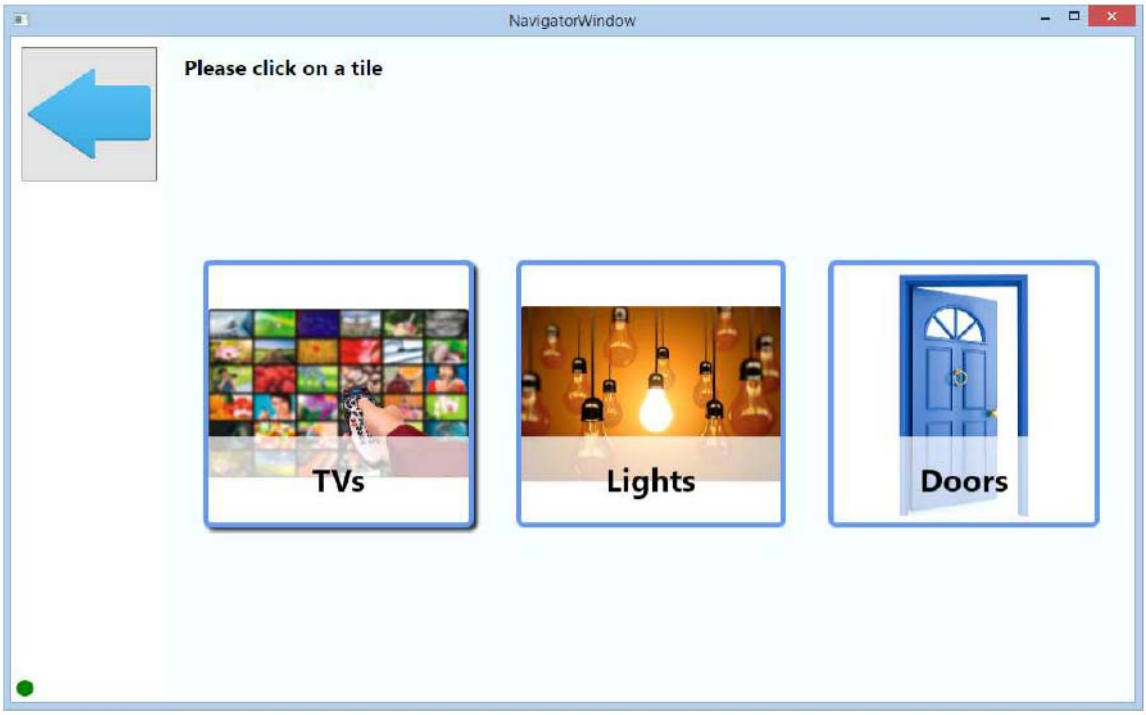


Figure 76: The Home Control submenu of the main menu application regarding the control of different devices in the surrounding environment.

Figure 76 shows the home control submenu of the main menu application containing the items that trigger the respective applications regarding the control of various home aspects such as the TVs, the lights and the doors of the house while Figure 77 shows the televisions screen of the TV application containing all the televisions around the house. The user can select the desired TV in order to navigate to the TV control screen.

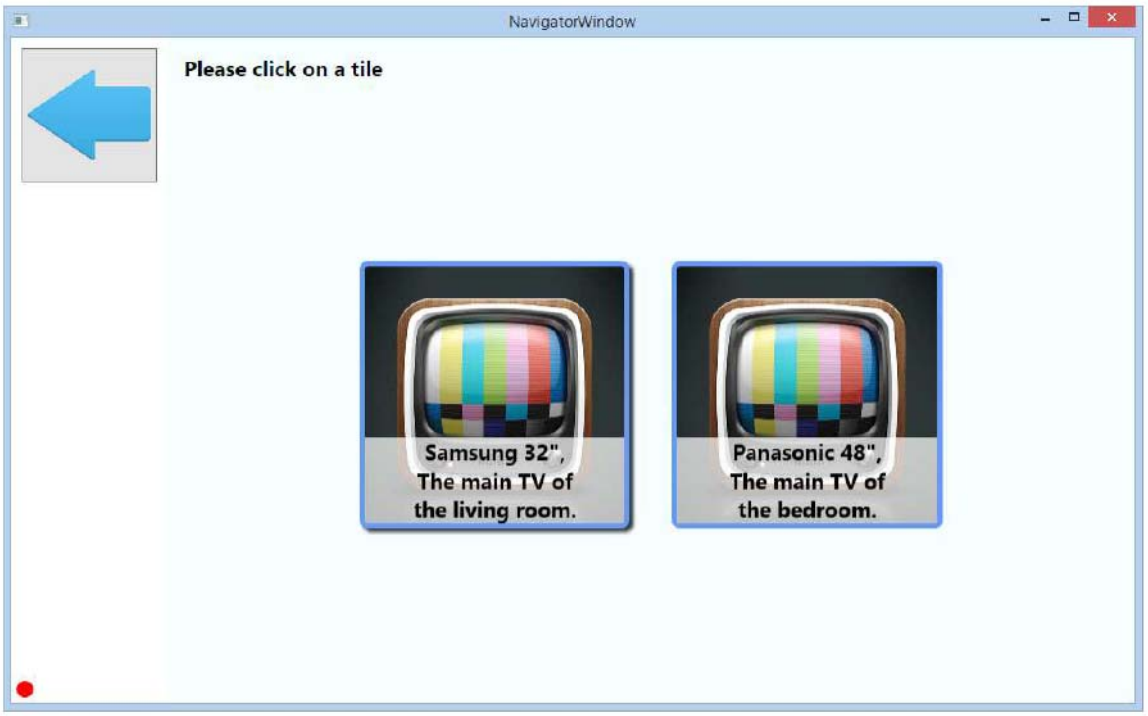


Figure 77: The televisions screen of the TV control application

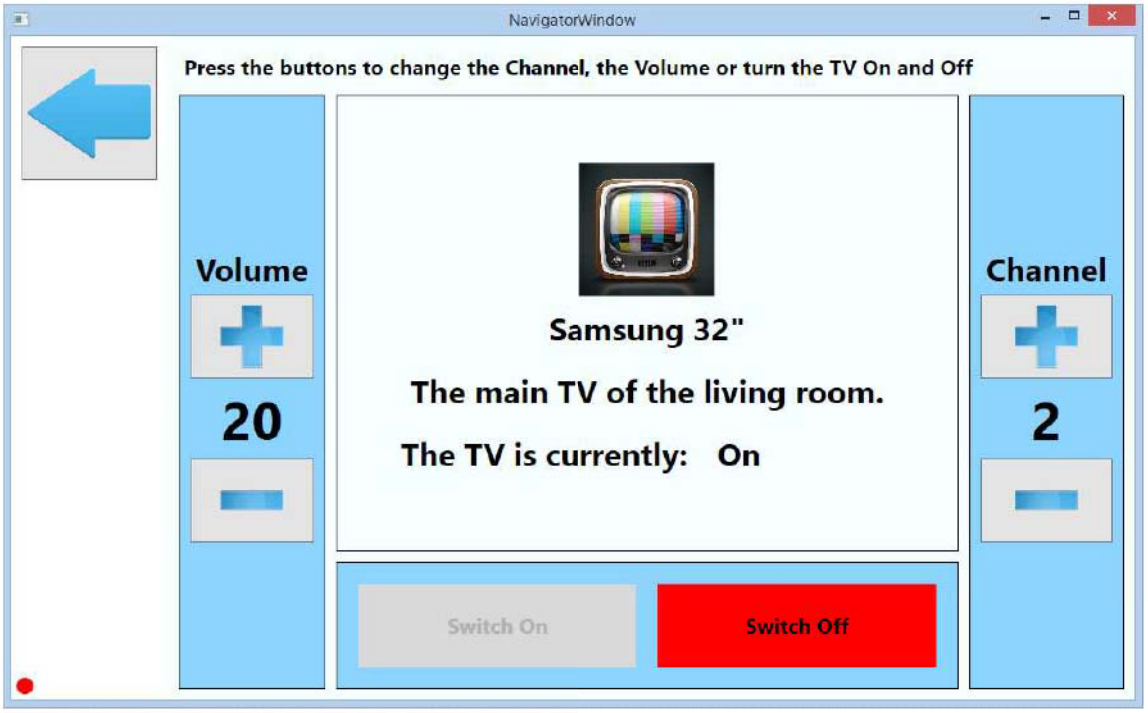


Figure 78: The TV control application

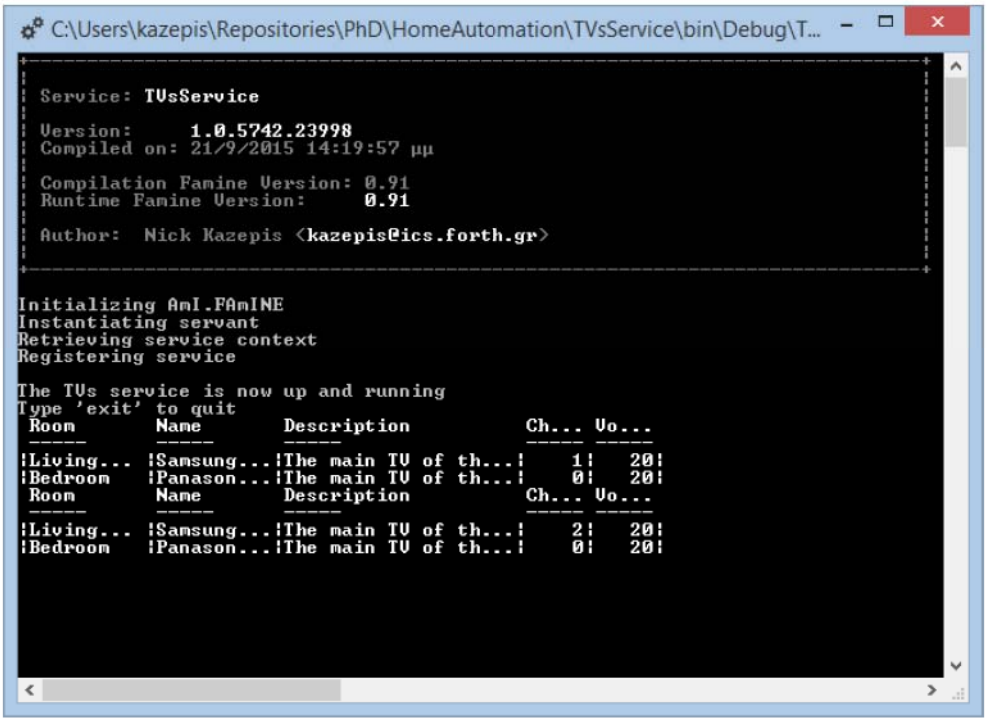


Figure 79: FAmINE service exposing the functionality for remotely controlling a TV

Figure 78 shows the TV control dialogue of the TV control application. When the user selects the TV that he/she wants to control, the TV control dialogue appears that gives the users the functionality for remotely controlling the selected TV. The user can turn the TV on or off as well as change the level of the volume and the currently displaying channel. By pressing the corresponding buttons, the underlying FAmINE service gets the function calls and emits the values of the changed properties in the form of events. These events are being captured by the UI to refresh the currently displaying values. The whole process is completely transparent to the programmer as demonstrated in the previous chapter.

```
#include <ami.idl>

module HomeAutomation {
    interface TVsService {

        // Types
        struct TVEntry {
            string room;
            string name;
            string description;
            ami::OctetSeq picture;
            long lastChannel;
            long channel;
            long volume;
        };
        typedef sequence<TVEntry> TVSeq;

        // Variables
        readonly attribute string ServiceDescription;

        // Methods
        TVSeq GetRoomTVs (in string room);
        TVSeq GetAllTVs ();

        void SwitchOn (in string room, in string tv);
        void SwitchOff (in string room, in string tv);

        void VolumeUp (in string room, in string tv);
        void VolumeDown (in string room, in string tv);

        void ChannelUp (in string room, in string tv);
        void ChannelDown (in string room, in string tv);
        void GoToChannel (in string room, in string tv, in long targetChannel);


        long GetChannel (in string room, in string tv);
        long GetVolume (in string room, in string tv);

        // Events
        void Event_TVOn  (in string room, in string tv, in long channel);
        void Event_TVOff (in string room, in string tv);
        void Event_TVChannelChange (in string room, in string tv, in long oldChannel, in long newChannel);
        void Event_TVVolumeChange (in string room, in string tv, in long volume);
    };
};
```

Figure 80: The CORBA idl description of the sample TV service application

Figure 79 shows the TV service running on a remote machine. The service prints on the console the respective messages every time a TV is turned on or off and every time the volume or the channel that TV is displaying changes. The interface that the TV service implements can be seen in the Figure 80. The service uses various types, variables methods and events. The service is capable of delivering the properties of all available TV sets throughout the house. As a result, the dialogue that is shown in Figure 77 can be automatically generated using the framework's option presenter dialogue screen. The events that can be seen in the Figure 80 have been implemented as shown in 5.9.1, while the calling of the respective functions has been realized using the FAmINE blackbox that was developed as part of the framework. Finally, methods like the GetAllTVs() that return a sequence of custom type declared object instances

are automatically converted into a list of dynamic type variables so that the developer is presented with a list of dynamic objects, where each object contains the declared parameters as properties. For example, in the case of GetAllTVs, the developer gets a list of dynamic objects, each of which contains a property for the room name, a property for the channel, a property for the picture etc.

8 Summary, Discussion and Future Work

8.1 Summary and Discussion

The work conducted in the context of this thesis has addressed the creation of elderly friendly multimodal interactive applications in the context of human robot interaction between the users and household assistive robotic platforms deployed in ambient intelligence environments. To this end, an in-depth study of human robot interaction involving elderly users has been conducted, in order to acquire valuable insight about the utility and usability that multiple interaction modalities and ambient intelligence environments can offer. In the same context, the adaptation needs and potential of such systems has been investigated. The acquired knowledge was extended to address the diverse needs of the target user population and the vast heterogeneity of robotic systems and ambient intelligence environments. In this context, the employment of ambient intelligence environment by domestic assistive robotic platforms was explored. The user-robot-environment triad was studied in depth to identify and provide the means to the end of natural interaction. In this context, the proposed work offers the tools and technologies to cover the identified needs towards bridging the gap between elderly users, household assistive robots and ambient intelligence environments.

The current study focuses on creating a universal solution to be used as the corner stone for building multimodal, elderly friendly, interactive applications that target household robots for elderly users. This thesis provides developers with the necessary technologies, tools and building blocks for creating easy to use elderly-friendly multimodal applications in AAL environments, with particular focus on robotic platforms, thus increasing their level of adaptation to users' needs, and, as a consequence, users' acceptance of these technologies. Using the proposed framework, makes these applications inherently friendly to the elder users, while adapting to their needs, the surrounding environment and the context of use. This results in a smooth learning curve, providing increased levels of user satisfaction. Moreover, the use of the proposed framework introduces new interaction modalities

such as voice and gestures, enriching and simplifying the interaction experience of the applications. The key contribution of the current work is summed up in the following paragraphs.

8.2 Contribution of this thesis

As discussed at several points in this thesis, the scope of addressing users, domestic robotic platforms and the ambient intelligence environment from the perspective of interaction was a very challenging task. To achieve the goals set by this research work, several new directions of research were exploited and existing research was reconsidered under different perspectives. Furthermore, it was required to address ambitious goals such as, for example, serving optimally both developers/designers and end-users. This was challenging as developers from the programmer's perspective focus on fast, robust and rapid to develop frameworks paying less attention to the end user. The burden of ensuring that good decisions are made falls in the hands of designers. However, designers especially in SMEs have limited resources at their disposal while in some cases they have to make bad decisions in favor of reduced development costs. Nevertheless, the end users' expectations and their overall satisfaction of using the system is a major success criterion. Feeling like having our feet in different boats, this research work contributes significantly to computer science by allowing on the one hand to develop adaptive multimodal applications for robotic platforms without having to mess with the mechanics of multimodality and adaptation while on the other hand providing the best to the end users from an HRI perspective along with ensuring the optimum cooperation and interconnection of applications with the robotic platform itself and the smart environment hosting it.

To achieve this contribution, research was conducted in several directions. The rest of this section summarizes the most important computer science domains were significant contributions were produced.

Decision logic for HRI

Taking advantage of the scripting functionalities of ACTA, a general purpose finite state machine (FSM) description language for ACTivity Analysis [139], this work extends the runtime infrastructure of the language in order to enable the

multimodality integration and achieve transparent and independent application state management for concurrently active applications. Furthermore, this approach facilitates the leveraging of the application logic to a higher level where changes can be easily made with minimal source code meddling. The fine tuning of the applications and the rearrangement of the application screens can be made by editing the ACTA script which is in human readable format, hence reducing the changes in the application source code which in turn reduces the margin for error.

Adaptation

The FIRMA framework offers a collection of ready-made controls, framework elements and UI dialogue sequences (in the form of adaptive component hierarchies) that contribute to the reduction of the necessary code to build an interactive adaptive application, hence reducing the margin for error and the required development time. Using the FIRMA framework happens in a transparent to the developer way while the framework itself carries out the administrative and decision making processes. Furthermore, in contradiction to the existing research framework in the domain of Unified User Interfaces, FIRMA radically simplifies the integration of novel controls to the framework by relying not on polymorphic task hierarchies but on adaptive style hierarchies thus allowing the designer/developer to only provide alternative styles for alternative representations. Additionally, through modern designer-oriented tools this process can also be done by designers thus totally removing developers from the adaptation loop. Moreover, this approach achieves transparent adaptation of the created dialogues based on the user's needs and preferences as well as transparent adaptivity to the context of the interaction.

Styling multimodal applications

This work supports adaptive cascading style hierarchies. This approach allows designers/developers to develop alternative styles for their products without having to alter the functionality and logic of the framework based applications thus minimizing the development effort without the requirement for rebuilding application components. Using FIRMA, offers a strategic advantage by assisting to the development of a corporate look and feel.

Globalization & localization

An important aspect of commercial exploitation of any product in the market is certainly the support for adapting also to cultural characteristics of each targeted user. While this is partially address by the adaption logic of the system it should be also facilitated at a lexical and syntactic level. Applications created using FIRMA are globalization compatible as the necessary tools for localizing the applications to different languages, cultures and locales are inherently build into the framework. Furthermore the localization files are automatically generated by the system without the need for the developer to even be aware of their generation. The needed appropriate adaptations to these automatically generated localization files can be performed offline and without the need to recompile the source code of the developed applications, thus offering the possibility for this adaptation to be done in the premises of the resellers of the robotic platform and applications.

Inherent multimodality integration

Speech recognition and gesture recognition modalities have been inherently incorporated into the framework and adapted to comply with the activation of the different framework elements that were built, hence exempting the developer from having to integrate each modality in a different way. The interaction modality integration process has been offered as part of the framework, and the parameterization of the modality functionality has been supported at a higher level by means of application logic scripts written using the ACTA scripting language. This mechanism reserves space for future extensions of the framework thus achieving scalability ensuring that in the case of emergence of new interaction modalities the system will scale successfully to meet new demands and new challenges.

Pervasive middleware integration

FORTH's Ambient Intelligence Network Environment (FAmINE) has been incorporated into the framework, allowing the robot to achieve augmented cognition through the environment (the robot can receive events from the environment and respond accordingly) and to adapt applications (running both on the robot and the environment) based on dynamically changing environment factors. This approach offers the potential to create more complex robot behaviors while offering increased functionality that leads to higher levels of subjective user satisfaction and acceptance

of these technologies. Furthermore the integration layer has been developed as a plugin to the framework itself thus enabling the development of different plugins for different middlewares in different environments (e.g. OpenIoT).

ROS operating system integration

The framework provides inherent support for the Robot Operation System (ROS) which is the standard operating system for the vast majority of existing robots and comprises a set of software libraries and tools that help to support the robot applications on the hardware side. This approach enables the usage of the proposed framework for the Human-Robot interaction part on any existing and future platform that uses ROS. The bridging of the linux side ROS operating system with the reasoning capabilities offered by the application logic can provide the base for building applications that take advantage of the robot's hardware capabilities and contribute to more complex behavioral scenarios.

HRI accessibility

This research work envisions and provides the means towards the establishment of HRI accessibility concerning the Human-Robot Interaction field, since the implemented approaches can be tweaked and extended to suit the needs of disabled people as well. Such extensions can happen transparently to the FIRMA's core functionality requiring only the addition of controls, styles and logic.

Contribution through the European funded HORIZON 2020 project RAMCIP

This research work constitutes the primary platform for the development of the user interfaces for the Human Robot Interaction part of the RAMCIP project (a three-year research project funded by the European Commission under the HORIZON 2020 programme which started in January 2015 to support elderly people with Mild Cognitive Impairment). The validity and the effectiveness of the framework will be tested both in the lab and in real life scenarios in the pilot trials of the project which will take place simultaneously in two different European cities over a time span of about 6 months.

8.3 Future work

Two main directions of further work are anticipated in a path towards supporting the fruition of domestic assistive platforms for the elderly in Ambient Intelligence Environments. The first is the further enrichment and development of the system into a mature product and the second is the adaptation of the framework to cover the needs of different user categories and impairments as well as their families' and caregivers'.

The first direction concerns the improvement of developed infrastructure and applications and their evolution from an in-vitro prototype to a mature software product. Towards this end, a number of planned improvements of the application are stemming from the conducted heuristic evaluation and the development process of the two aforementioned application case studies. To this end, there is a number of less critical usability errors identified using the expert based evaluation of the applications that should be addressed together with the results that are going to be produced by the user based evaluation. These results are important because they represent potential issues of the end-users that have negative impact on their experience with the system. Additionally, the integration of the proposed architectural framework and applications in the RAMCIP robotic platform and in a more mature (currently being equipped) simulation space within FORTH's AmI facility will pose several requirements in terms of integrating new sensors and automation mechanism to the system. This is going to generate new requirements both in terms of integrating new technology but also facilitating the input generated by this technology for creating smarter and more adaptive system behaviours.

To this end, along with the results of the heuristic UI evaluation, a number of functional improvements have already been identified regarding the currently available modalities as well as the overall system functioning in terms of performance, offered functionality and ease of use regarding the software developers.

8.3.1 Performance improvements

Although the framework is robust, scalable and overall stable, a number of improvements have been identified regarding the performance of the system. In this

scope, several subcomponents of the framework will be optimized and fine-tuned to achieve maximum performance. For example, the ACTA runtime module will be tweaked to queue consequent ACTA-triggered function calls in order to minimize the evaluation time per ruleset iteration. This will achieve the maximum performance regarding the rule engine that is being used for the reasoning capabilities of the system. Furthermore, in the context of memory optimizations, the different applications' dialogues are going to be tweaked to be able to provide their state and data in a standalone detachable object that could be detached from the dialogue and stored while the dialogue would be destroyed to release reserved resources. This would have a positive impact on the memory footprint of the whole system without any loss in performance since no data sacrifice would take place that would require data regeneration.

8.3.2 Functionality improvements

In the context of functionality, a number of improvements have been identified regarding the different available modalities. Regarding the speech recognition modality, the speech recognition engine will be fine-tuned to be able to run analyses on the provided user grammars, identify potentially problematic phrases and adjust the recognition threshold at runtime based on the phrase that would be recognized. Furthermore, the addition of a touch based gesture recognition modality to support touch gestures such as "pinch to zoom", "swipe to navigate", etc. will be considered. Finally, the provision for additional modalities will be examined. Additional modalities will give the opportunity for even smarter behavioral models since additional user groups will be supported. For example, the addition of a hardware switches modality, a mouth actuators modality etc. would make it possible for users with mobility impairments to use the system. To this end, the support for scanning between all available display components, controls and elements will be examined as well.

Furthermore, the possibility to split the screen into multiple separate areas of usable real estate will be examined. The potential to display more than one application status or actions / notifications simultaneously will be explored. In this scope, this approach will enable more than one application to be visible at a time while different applications will be able to ask for different parts of the screen to display meaningful

information. For example, the screen will be able to display the status of the current robot action as well as the next action which is going to execute or display content from different applications such as the time (from the clock application) and the TV control dialogue (from the TV automation application). Finally, this approach will be closely examined with elderly end-users in order to evaluate and validate its feasibility potential in terms of acceptance, since the golden mean between providing an efficient and usable system and displaying too little or too much information must be struck.

8.3.3 Developer ease of use improvements

Last but not least, a number of improvements regarding the developers that will use the FIRMA framework to build multimodal adaptable and adaptive elderly friendly applications will be provided. The necessary Visual Studio automation projects and class types are going to be provided in order for the framework to be fully integrated into the Visual Studio IDE. Developers will be able to select the correct type of projects and include the correct type of user controls into their applications that already incorporate the framework's design styles. Furthermore, the provision of a standalone SRGS editor for developing speech recognition grammars and fine tuning or translating already existing ones will be examined. Moreover, a standalone tool for managing the automatically generated translation files of the application will be examined as well. Additionally, the implementation of a strictly formed communication protocol between the ROS operating system and services offered by the ambient intelligence environment will have positive impact by providing formal communication and defining functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the underlying interface. Finally, base classes provided in the framework will be tweaked to be made responsible for taking the latest ACTA script and transcribing into an XML rules file that can be used directly by the application for reasoning purposes in order to eliminate synchronization issues or the need to keep additional rule files.

All the aforementioned improvements and additions will create new possibilities regarding the adaptation and the adoption of the FIRMA framework. This future work

will open new horizons to the fruition of assistive robots in ambient intelligence environments. The FIRMA framework will become an even more complete universal solution that will be suitable to be used as the corner stone for building multimodal, elderly friendly, interactive applications that target household robots for a diverse target population. Different user groups can be targeted and the framework will be capable of adapting to their special needs, impairments, disabilities and requirements as well as to the families and relatives of the primary users. The presented framework is going to be used in the context of the European funded Horizon 2020 RAMCIP project that targets elderly users with mild cognitive impairment as well as users who are on the first stage of agnostic dementia. This is going to be an invaluable lesson towards additional functionality necessities, improvements and future work that will have a significant impact on the maturity of the presented work.

9 Publications

Parts of the current thesis have been submitted to international conferences and journals according to the following list:

- Kazepis N., Antona M., Stephanides C. (2015, submitted) "FIRMA: A Development Framework for Elderly-Friendly Interactive Multimodal Applications for Assistive Robots", Ninth International Conference on Advances in Computer-Human Interaction, ACHI 2016, April 24 - 28, 2016 - Venice, Italy
- Kazepis N., Antona M., Stephanides C, (2015, submitted) Employing pervasive middlewares to achieve augmented context-awareness of robotic assistants in intelligent environments, Journal of Ambient Intelligence and Smart Environments (JAISE), IOS Press

10 References

[1] W. Cox, L. Y. Abramson, P. G. Devine and S. D. Hollon, "Stereotypes, Prejudice, and Depression: The Integrated Perspective," *Perspectives on Psychological Science,* 2012.

[2] H. F. Paffrath, "The timeliness of ideological criticism in the media age," *Critique of technology and media education.,* 1998.

[3] M. Alaoui, M. Lewkowicz and A. Seffah, "Increasing elderly social relationships through TV-based services," in *IHI '12: Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, 2012.

[4] D. López-de-Ipiña, S. Blanco, X. Laiseca and I. Díaz-de-Sarralde, "ElderCare: An Interactive TV-based Ambient Assisted Living Platform," in *Actas del II International Workshop of Ambient Assisted Living (IWAAL 2010)*, Valencia, Spain, 2010.

[5] C. Schroeter, S. Mueller, M. Volkhardt, E. Einhorn, C. Huijnen, H. van den Heuvel, A. van Berlo, A. Bley and H.-M. Gross, "Realization and User Evaluation of a Companion Robot for People with Mild Cognitive Impairments," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA 2013), Karlsruhe, Germany, pp. 1145-1151, IEEE 2013*, Karlsruhe, Germany, 2013.

[6] M. Vincze, *HOBBIT - Towards a robot for Aging Well - Workshop on Human Robot Interaction (HRI) for Assistance Robots and Industrial Robots, May 6-10,* ICRA 2013, Karlsruhe, 2013.

[7] V. K. Emery, P. J. Edwards, J. A. Jacko, K. P. Moloney, L. Barnard, T. Kongnakorn, F. Sainfort and I. U. Scott, "Toward Achieving Universal Usability for Older Adults Through Multimodal Feedback," in *Proceedings of the 2003 conference on Universal usability - CUU '03*, 2003.

[8] D. Warnock, M. McGee-Lennon and S. Brewster, "Multiple notification modalities and older users," in *CHI '13: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[9] J. Blecharz and M. Siekańska, "Psychologiczne aspekty starzenia się i starości.," *Fizjologia starzenia się. Profilaktyka i rehabilitacja. A. Marchewka, Z. Dąbrowski, J.A. Żołądź. Warszawa: Wydawnictwo Naukowe PWN,* pp. 48-59, 2012.

[10] J. Rowe and R. Kahn, "Successful aging and disease prevention.," *Adv Ren Replace Ther.,* vol. 7, no. 1, pp. 70-77, 2000.

[11] J. Rowe and R. Kahn, "Successful aging.," *Gerontologist.,* vol. 37, no. 4, pp. 433-440, 1997.

[12] *World Health Organization and Alzheimer's Diseases International: Dementia: a public health priority. Geneva: WHO Document Production Services.,* 2012.

[13] *World Health Organization: Good health adds life to years. Global brief for World Health Day. Geneva: WHO Document Production Services. 2012b.,* 2012.

[14] W. J. C. M. K. H. D. S. R. A.-S. Wu Y-H, "Acceptance of an assistive robot in older adults: a mixed-method study of human–robot interaction over a 1-month period in the Living Lab setting.," *Clinical Interventions in Aging,* vol. 9, pp. 801-811, 2014.

[15] M. S. B. S. P. K. R. S. E. J. Y. K. M. B. K. J. M. C. M. D. S. N. W. M. Chao L.L., "Evidence of neurodegeneration in brains of older adults who do not yet fulfill MCI criteria.," *Neurobiology of Aging,* vol. 31, pp. 368-377, 2010.

[16] L. A. M. M. W. C. E. D. S. L. R. B. A. D. J. O. R. S. S. L. A. T. S. B. D. B. N. M. L. M. R. A. S. Z. Z. A. M. R.-. T. S. Cook I.A., "Cognitive and physiologic correlates of subclinical structural brain disease in elderly healthy control subjects.," *Archives of Neurology,* vol. 59, 2002.

[17] A. M. R.-H. S. Hensel A., "Measuring cognitive change in older adults. Do reliable change indices of the SIDAM predict dementia?," *Journal of Neurology,* vol. 254, pp. 1359-1365, 2007.

[18] R. W. F. D. Reischies F.M., "Brain atrophy parameters of very old subjects in a population – based sample with and without dementia syndrome.," *European Archive of Psychiatry and Clinical Neurosciences,* vol. 251, pp. 99-104, 2001.

[19] E. Goldberg, "The Wisdom Paradox: How Your Mind Can Grow Stronger as Your Brain Grows Older.," *Free Press,* 2005.

[20] Alzheimer Disease International, "World Alzheimer Report 2015: The Global Impact of Dementia," [Online]. Available: http://www.alz.co.uk/research/world-report-2015.

[21] L. M. H., "Ośrodkowy i obwodowy układ nerwowy – starzenie się fizjologiczne i profilaktyka.," *Fizjologia starzenia się. Profilaktyka i rehabilitacja. A.Marchewka, Z. Dąbrowski, J.A. Żołądź. Warszawa: Wydawnictwo Naukowe PWN,* pp. 102- 115, 2012.

[22] S. M. Blecharz J., "Psychologiczne aspekty starzenia się i starości. In: Fizjologia starzenia się. Profilaktyka i rehabilitacja. A. Marchewka, Z. Dąbrowski, J.A. Żołądź. Warszawa: Wydawnictwo Naukowe PWN," pp. 48-59, 2012.

[23] A. Jacobs and J. Pierson, "Walking the Interface: Domestication Reconsidered," *Brave New Interfaces,* pp. 205-215, 2007.

[24] P. De Hert and E. Mantovani, "Intelligent User Interface. Working Document of the Project "SENIOR: Social Ethical and Privacy Needs in ICT for Older Citizens: A Dialogue Roadmap"," 2008.

[25] S. Adam, K. Ssamula Mukasa, K. Breiner and M. Trapp, "An apartment-based metaphor for intuitive interaction with ambient assisted living applications," in *BCS-HCI '08: Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 1*, 2008.

[26] A. D'Andrea, A. D'Ulizia, F. Ferri and P. Grifoni, "A multimodal pervasive framework for ambient assisted living," in *PETRA '09: Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, 2009.

[27] T. A. Majchrzak, A. Jakubiec, M. Lablans and F. Ückert, "Towards better social integration through mobile web 2.0 ambient assisted living devices," in *SAC '11: Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011.

[28] E. Aarts and J. Encarnacao, True Visions. The Emergence of Ambient Intelligence. ISBN 978-3-540-28972-2, Springer, 2008.

[29] A. Valli, "The design of natural interaction. Multimedia Tools and Applications.," vol. 38, no. 3, pp. 295-305, 2008.

[30] E. Aarts and B. de Ruyter, "New research perspectives on Ambient Intelligence.," *Journal of Ambient Intelligence and Smart Environments.,* vol. 1, no. 1, pp. 5-14, 2009.

[31] F. Zhou, H. Been-Lirn Duh and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR.," *ISMAR 2008,* pp. 193-202, 2008.

[32] C. Ramos, "Ambient intelligence - a state of the art from artificial intelligence perspective, In Proceedings of the aritficial intelligence 13th Portuguese conference on Progress in artificial intelligence (EPIA'07)," Berlin, Heidelberg, 2007.

[33] M. Alcañiz and B. Rey, "New Technologies For Ambient Intelligence, IOS Press," 2005. [Online]. Available: http://www.ambientintelligence.org.

[34] N. Streitz, T. Prante, C. Roecker, D. Van Alphen, R. Stenzel, C. Magerkurth, S. Lahlou, V. Nosulenko, F. Jegou, F. Sonder and D. Plewe, "Smart artefacts as affordances for awareness in distributed teams. In The disappearing computer,

Norbert Streitz, Achilles Kameas, and Irene Mavrommati (Eds). Lecture Notes In Computer Science," vol. 4500, pp. 3-29, 2007.

[35] H. Nicolau and J. Jorge, "Elderly text-entry performance on touchscreens," in *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility - ASSETS '12*, 2012.

[36] M. Kobayashi, A. Hiyama and T. Miura, "Elderly user evaluation of mobile touchscreen interactions," in *Human-Computer Interaction – INTERACT 2011 Lecture Notes in Computer Science*, 2011.

[37] N. Hollinworth and F. Hwang, "Investigating familiar interactions to help older adults learn computer applications more easily," in *BCS-HCI '11 Proceedings of the 25th BCS Conference on Human-Computer Interaction*, 2011.

[38] C. Wacharamanotham, J. Hurtmanns, A. Mertens, M. Kronenbuerger, C. Schlick and J. Borchers, "Evaluating swabbing: a touchscreen input method for elderly users with tremor," in *Proceedings CHI '11 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011.

[39] C. Stößel and L. Blessing, "Mobile device interaction gestures for older users," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*, 2010.

[40] C. Leonardi, A. Albertini, F. Pianesi and M. Zancanaro, "An exploratory study of a touch-based gestural interface for elderly," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*, 2010.

[41] J. A. Jacko, n. U. IScott, F. Sainfort, K. P. Moloney, T. Kongnakorn, B. S. Zorich and V. K. Emery, "Effects of Multimodal Feedback on the Performance of Older Adults with Normal and Impaired Vision," in *Universal Access Theoretical Perspectives, Practice, and Experience Lecture Notes in Computer Science*, 2003.

[42] Y. Li, "VoiceLink: A Speech Interface For Responsive Media," 2002.

[43] A. Postma, "Detection of errors during speech production: a review of speech monitoring models," Psychological Laboratory, Helmholtz Institute, Utrecht University, Heidelberglaan.

[44] A. B. J. D. M. G. K. G. V. G. D. P. a. O. S. Mahesh S. Raisinghani, "Ambient Intelligence: Changing Forms of Human-Computer Interaction and their Social Implications," The German International Graduate School of Management and Administration (GISMA), Purdue University, West Lafayette, IN, USA.

[45] M. H. D. L. D. Lezak, "Neuropsychological assessment," Oxford University Press, New York, 2004.

[46] K. Oka, Y. Sato and H. Koike, "Real-time Tracking of Multiple Fingertips and Gesture Recognition for Augmented Desk Interface Systems".

[47] O. K. K. M. M. S. S. Y. K. H. Nakanishi Y, "Narrative Hand: Applying a fast finger-tracking system for media art".

[48] O. J, "Computer Vision Based Analysis of Non-verbal Information in HCI".

[49] H. Kim and W. D. Fellner, "Interaction with Hand Gesture for a Back-Projection Wall".

[50] T. Ohno, M. N. and S. Kawato, "Just Blink Your Eyes: A Head-Free Gaze Tracking System".

[51] T. Ohno and N. Mukawa, "A Free-head, Simple Calibration, Gaze Tracking System That Enables Gaze-Based Interaction".

[52] T. Ohno, N. Mukawa and A. Yoshikawa, "FreeGaze: A Gaze Tracking System for Everyday Gaze Interaction".

[53] Universal Design Organization, [Online]. Available: http://www.universaldesign.org/universaldesign3.htm.

[54] C. Stephanidis, "User Interfaces for All: New perspectives into Human-Computer Interaction," in *C. Stephanidis (Ed.), User Interfaces for All - Concepts,*

Methods, and Tools, Mahwah, NJ, Lawrence Erlbaum Associates (ISBN 0-8058-2967-9), 2001, pp. 3-17.

[55] C. Stephanidis, "Adaptive techniques for Universal Access," *User Modelling and User Adapted Interaction International Journal,* vol. 11, no. (1/2), pp. 159-179, 2001.

[56] C. Stephanidis, G. Salvendy, D. Akoumianakis, N. Bevan, J. Brewer, P. Emiliani, A. Galetsas, S. Haataja, I. Iakovidis, J. Jacko, P. Jenkins, A. Karshmer, P. Korn, A. Marcus, H. Murphy, C. Stary, G. Vanderheiden, G. Weber and W. Ziegler, "Toward an Information Society for All: An International R&D Agenda," *International Journal of Human-Computer Interaction,* pp. 107-134, 1998.

[57] J.-P. Leuteritz, H. Widlroither, A. Mourouzis, M. Panou, M. Antona and A. Leonidis, "Development of Open Platform based Adaptive HCI Concepts for Elderly Users," in *C. Stephanidis (Ed). Universal Access in Human Computer Interaction - Intelligent and Ubiquitous Interaction Environments - Volume 6 of the Proceedigns of HCI International 2009.*, San Diego, CA, 2009.

[58] A. Leonidis, M. Antona and C. Stephanidis, "Rapid Prototyping of Adaptable User Interfaces," *International Journal of Human-Computer Interaction,* vol. 28, no. 4, pp. 213-235, 2012.

[59] M. Foukarakis, A. Leonidis, I. Adami, M. Antona and C. Stephanidis, "An Adaptable Card Game for Older Users.," in *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments - PETRA*, Crete, Greece, 2011.

[60] G. Zimmermann, A. Stratmann, D. Reeß and T. Glaser, "Patterns for User Interface Adaptations: Towards Runtime Adaptations for Improving the Usability of Web Forms for Elderly Users," in *Jia Zhou and Gavriel Salvendy (Eds.) Human Aspects of IT for the Aged Population. Design for Aging*, Los Angeles, 2015.

[61] M. Peissner and R. Edlin-White, "User Control in Adaptive User Interfaces for Accessibility," in *Human-Computer Interaction – INTERACT 2013*, Cape Town, South Africa, 2013.

[62] MyUI Consortium , "MyUI: Mainstreaming Accessibility through," 2010.

[63] K. Dautenhahn, Human-Robot Interaction. In: Soegaard, Mads and Dam, Rikke Friis (eds.). "The Encyclopedia of Human-Computer Interaction, 2nd Ed.". Aarhus, Denmark: The Interaction Design Foundation., 2014.

[64] M. A. Goodrich and A. C. Schultz, "Human–Robot Interaction: A Survey," *Foundations and Trends in Human–Computer Interaction,* vol. 1, no. 3, pp. 203-275, 2007.

[65] W. M. a. J. M. M. Betke, "Active detection of eye scleras in real time," in *IEEE Workshop on Human Modeling, Analysis and Synthesis*, Hilton Head, South Carolina, , June 2000.

[66] E. Drumwright, O. C. Jenkins and M. J. Mataric, "Exemplar-based primitives for humanoid movement classification and control," in *IEEE International Conference on Robotics and Automation*, Apr 2004.

[67] E. Horvitz and T. Paek., "Harnessing models of users' goals to mediate clarification dialog in spoken language systems," in *International Conference on User Modeling*, 2010.

[68] D. Wang and S. Narayanan, "An acoustic measure for word prominence in spontaneous speech," in *IEEE Transactions on Speech, Audio and Language Processing*, 2007.

[69] G. Rizzolatti and A. MA., "Language within our grasp," *Trends in Neurosciences,* vol. 21, no. 5, pp. 188-194, 1998.

[70] B. Scassellati, "Investigating models of social development using a humanoid robot," in *Internanational Joint Conference on Neural Networks*, 2003.

[71] C. Breazeal, A. Edsinger, P. Fitzpatrick and B. Scassellati, "Active vision for sociable robots on Man, Cybernetics and Systems," in *IEEE Transactions*, 2001.

[72] C. Breazeal., "Infant-like social interactions between a robot and a human caretaker," in *Adaptive Behavior*, 2000.

[73] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki and K. Fujimura, "The intelligent ASIMO: system overview and integration.," in *In International Conference on Intelligent Robots and System*, 2002.

[74] M. Hunke and A. Waibel, "Face locating and tracking for human-computer interaction," in *Conference on Signals, Systems and Computers*, Pacific Grove, CA.

[75] M. C. Shin, K. I. Chang and L. V. Tsap, "Does colorspace transformation make any difference on skin detection?," [Online]. Available: http://citeseer.nj.nec.com/542214.html.

[76] C. Breazeal, G. Hoffman and A. Lockerd, "Teaching and working with robots as a collaboration," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[77] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte and D. Schulz, "MINERVA: A second-generation museum tour-guide robot.," in *IEEE International Conference on Robotics and Automation*, 1999.

[78] F. Michaud, J.-F. Laplante, H. Larouche, A. Duquette, S. Caron, D. Letourneau and P. Masson, "Autonomous spherical mobile robot for child-development studies," in *IEEE Transactions on Systems, Man and Cybernetics*, 2005.

[79] K. Kim, S. Bang and S. Kim, "Emotion recognition system using short-term monitoring of physiological signals.," *Medical and Biological Engineering and Computing,* vol. 42, no. 3, pp. 419-427, 2004.

[80] A. Mohan and R. Picard., "Health 0: A new health and lifestyle management paradigm," *Stud Health Technol Inform,* vol. 108, pp. 43-48, 2004.

[81] D. B. Shalom, S. H. Mostofsky, R. L. Hazlett, Goldberg, R. J. Landa, Y. Faran, D. R. McLeod and R. Hoehn-Saric, "Normal physiological emotions but differences in expression of conscious feelings in children with high-functioning autism," *Journal of Autism and Developmental Disorders,* vol. 36, no. 3, p. 295–400, 2006.

[82] E. Mower, D. Feil-Seifer, M. Mataríc and S. Narayanan, "Investigating implicit cues for user state estimation in human robot interaction.," in *International Conference on Human-Robot Interaction (HRI)*, 2007.

[83] T. Sowa and S. Kopp., "A cognitive model for the representation and processing of shape-related gestures," in *European Cognitive Science Conference (EuroCogSci03)*, New Jersey, 2003.

[84] N. Miller, O. Jenkins, M. Kallman and M. Mataric, "Motion capture from inertial sensing for unteth- ered humanoid teleoperation.," in *IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2004)*, Santa Monica, CA, 2004.

[85] J. Eriksson, M. Mataríc and C. Winstein, "Hands-off assistive robotics for post-stroke arm rehabilitation," in *International Conference on Rehabilitation Robotics*, Chicago, 2005.

[86] M. Stone and D. DeCarlo, "Crafting the illusion of meaning: Template-based specification of embodied conver- sational behavior," in *International Conference on Computer Animation and Social Agents*, 2003.

[87] A. Duquette, H. Mercier and F. Michaud, "Investigating the use of a mobile robotic toy as an imitation agent for children with autism.," in *International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, Paris, 2006.

[88] D. Grollman and O. Jenkins, "Learning robot soccer skills from demonstration," in *International Conference on Development and Learning (ICDL)*, London, 2007.

[89] M. Michalowski, S. Sabanovic and H. Kozima, "A dancing robot for rhythmic social interaction.," in *Conference on Human-Robot Interaction (HRI)*, Washington D.C., 2007.

[90] B. Mutlu, A. Krause, J. Forlizzi, C. Guestrin and J. Hodgins, "Robust, low-cost, non-intrusive recognition of seated postures.," in *20th ACM Symposium on User Interface Software and Technology*, Newport, RI., 2007.

[91] A. Kapoor and R. W. Picard., "Multimodal affect recognition in learning environments.," in *13th annual ACM international conference on Multimedia*, Singapore, 2005.

[92] S. Lang, M. Kleinehagenbrock, S. Hohenner, J. Fritsch, G. A. Fink and G. Sagerer, "Providing the basis for human-robot-interaction: A multi-modal attention system for a mobile robot.," in *International Conference on Multimodal Interfaces*, Vancouver, Canada, 2003.

[93] G. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal,* vol. 2, no. 2, pp. 12-21, 1998.

[94] J. Valin, F. Michaud, J. Rouat and D. Letourneau, "Robust sound source localization using a microphone array on a mobile robot. Intelligent Robots and Systems, 2003.(IROS 2003).," in *IEEE/RSJ International Conference on Robotics*, 2003.

[95] K. Mikolajczyk, C. Schmid and A. Zisserman, "Human Detection Based on a Probabilistic Assembly of Robust Part Detectors.," in *8th European Conference on Computer Vision*, Prague, Czech Republic, 2008.

[96] C. Stephanidis, "A European Ambient Intelligence Research Facility at FORTH – ICS," *ERCIM NEWS,* p. 31, 2006.

[97] F. Sadri, "Ambient intelligence: A survey," *Computing Surveys (CSUR) Volume 43 Issue 4,* October 2011.

[98] A. Gárate, N. Herrasti and A. López, "GENIO: an ambient intelligence application in home automation and entertainment environment," in *In Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies (sOc-EUSAI '05). ACM*, New York, 2005.

[99] D. Cook and S. Das, Smart Environments: Technologies, Protocols and Applications, John Wiley and Sons, 2004.

[100] A. Pounds-Cornish and A. Holmes, "The iDorm - A Practical Deployment of Grid Technology," in *Cluster Computing and the Grid. 2nd IEEE/ACM International Symposium on*, Brisbane, Australia, 2002.

[101] Wikipedia, "Assisted Living," [Online]. Available: http://en.wikipedia.org/wiki/Assisted_living.

[102] M. Pieper, M. Antona and U. (. Cortes, *Editorial: Ambient Assisted Living.,* ERCIM News, Special Theme: Ambient Assisted Living, 87, 18-19., 2011.

[103] ,. Riva, F. Vatalaro, F. Davide and M. Alcañiz, "The role of Ambient Intelligence in the Social Integration of the Elderly," 2005.

[104] I. Ilse Bierhoff, A. van Berlo, J. Abascal, B. Allen, A. Civit, K. Fellbaum, E. Kemppainen, N. Bitterman, D. Freitas and Kristiansson, "Smart Home Environment," in *Patrick R. W. Roe (Ed.) Towards an Inclusive Future: Impact and Wider Potential of Information and Communication Technologies*, COST219ter, 2006 , pp. 110-156.

[105] S. Oviatt, Multimodal interfaces. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, J. Jacko and A. Sears, Eds. Lawrence Erlbaum, Mahwah, NJ, 2003, p. 286–304.

[106] S. Oviatt and P. Cohen, "Perceptual user interfaces: multimodal interfaces that process what comes naturally.," *Communications of the ACM,* vol. 43, no. 3, pp. 45-53, 2000.

[107] S. Oviatt, "Multimodal interfaces for dynamic interactive maps.," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground (CHI '96) ACM*, New York, 1996.

[108] S. Oviatt, ,. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson and D. Ferro, "Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions.," *Human-Computer Interaction,* vol. 15, no. 4, pp. 263-322, 2000.

[109] P. Cohen and D. ,. C. J. McGee, "The efficiency of multimodal interaction for a map-based task.," in *Proceedings of the Sixth Conference on Applied Natural Language Processing Applied Natural Language Conferences. Association for Computational Linguistics,*, Morristown, NJ, 2000.

[110] J. L. Burke, M. S. Prewett, A. A. Gray, L. Yang, F. R. Stilson, M. D. Coovert, L. R. Elliot and E. Redden, "Comparing the effects of visual-auditory and visual-tactile feedback on user performance: a meta-analysis.," in *Proceedings of the 8th international Conference on Multimodal interfaces ICMI '06. ACM.*, New York, NY, 2006.

[111] I. Wechsung, N. A. B. and J. Hurtienne, "Multimodal Interaction: Intuitive, robust, preferred and senior-friendly?," 2009.

[112] A. Naumann, I. Wechsung and S. Möller, "Factors Influencing Modality Choice in Multimodal Applications. Perception in Multimodal Dialogue Systems,," in *4th IEEE Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems*, Berlin, 2008.

[113] J. Sturm and L. Boves, "Effective error recovery strategies for multimodal form - filling applications," *Speech Communication,* vol. 45, no. 3, pp. 289-303, 2005.

[114] L. Wu, S. Oviatt and P. Cohen, "Multimodal integration: A statistical view.," *IEEE Transactions on Multimedia,* vol. 1, no. 4, pp. 334-342, 1999.

[115] P. Cohen, M. Johnston, D. McGee, S. Oviatt, J. Pittman, I. Smith, L. Chen and J. Clow, "QuickSet: Multimodal interaction for distributed applications.," in *Proceedings of the Fifth ACM International Multimedia Conference*, New York, 1997.

[116] S. Oviatt, "Mutual disambiguation of recognition errors in a multimodal architecture.," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI'99) ACM*, New York, 1999.

[117] S. L. Smith and J. N. Mosier, "Guidelines for designing user interface software," in *Report No. MTR-10090, ESD-TR-86-278*, Bedford, 1986.

[118] Open Software Foundation, "OSF/Motif Style Guide, Revision 1.2," London: Prentice-Hall, 1993.

[119] Microsoft, "The WindowsTM interface guidelines for software design," Microsoft Press, Redmond, Washington, 1995.

[120] Apple Computers, "Macintosh human interface guidelines.," Addison Wesley, Reading, Massachusetts, 1992.

[121] IBM, "Object-Oriented Interface Design: IBM's Common User Access Guidelines, QUE.," 1992.

[122] S. H. K. R. M. Henninger, "A framework for developing experience-based usability guidelines.," in *Proc. of ACM Conf. DIS '95*, University of Michigan, August 23-25, 1995, 1995.

[123] S. L. C. F. C. Henninger, "Using organizational learning techniques to develop context-specific usability guidelines.," in *Proc. of Conf. DIS'97: Designing*

Interactive Systems: Process, Practices, Methods and Techniques, Amsterdam, 1997.

[124] J. Vanderdonckt, "Development Milestones towards a Tool for Working with Guidelines," *Interacting with Computers,* vol. 12, no. 2, pp. 81-118, 1999.

[125] L. M. Reeves, J. Lai, J. A. Larson, S. Oviatt, T. S. Balaji, S. Buisine and Q. Y. Wang, "Guidelines for multimodal user interface design," *Communications of the ACM,* vol. 47, no. 1, pp. 57-59, 2004.

[126] G. Cooper, Research in to Cognitive Load Theory and Instructional Design at UNSW, 1997.

[127] S. Kalyuga, P. Chandler and J. Sweller, "Managing split-attention and redundancy in multimedia instruction.," *Applied Cognitive Psychology,* vol. 13, pp. 351-371, 1999.

[128] K. Stanney, S. Samman, L. Reeves, K. Hale, W. Buff, C. Bowers, B. Goldiez, D. Lyons-Nicholson and S. Lackey, "A paradigm shift in interactive computing: Deriving multimodal design principles from behavioral and neurological foundations.".

[129] C. Wickens, Engineering Psychology and Human Performance (2nd ed)., New York: Harper Collins, 1992.

[130] D. McGee, P. Cohen and S. Oviatt, "Confirmation in multimodal systems.," in *In Proceedings of the International Joint Conference of the Association for Computational Linguistics and the International Committee on Computational Linguistics*, Montreal, Quebec, Canada, 1998.

[131] S. Oviatt, Multimodal interfaces. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications., Mahwah, NJ: J. Jacko and A. Sears, Eds. Lawrence Erlbaum, 2003, p. 286–304.

[132] M. K. C. B. G. J. F. T. F. J. L. R. W. a. A. Y. N. Quigley, "ROS: an open-source Robot Operating System.," *ICRA workshop on open source software,* vol. 3, no. 2, 2009.

[133] "The ROS wiki," [Online]. Available: http://wiki.ros.org/.

[134] Y. Georgalis, D. Grammenos and C. Stephanidis, "Middleware for ambient intelligence environments: Reviewing requirements and comcommunication technologies," *Universal Access in Human-Computer InInteraction. Intelligent and Ubiquitous Interaction Environments,* pp. 168-177, 2009.

[135] S. Kurkovsky, "Towards multimodal interfaces for the elderly," in *Multimodality in Mobile Computing and Mobile Devices: Methods for Adaptable Usability: Methods for Adaptable Usability*, IGI Global, 2009.

[136] D. Fischinger, P. Einramhof, W. Wohlkinger, K. Papoutsakis, P. Mayer, P. Panek, T. Körtner, S. Hofmann, A. Argyros, M. Vincze, W. A. and C. Gisinger, "Hobbit - The Mutual Care Robot," in *Proceedings of "Assistance and Service Robotics in a Human Environment" (ASROB 2013)*, Japan, 2013.

[137] S. Pheasant and C. M. Haslegrave, Bodyspace: Anthropometry, Ergonomics and the Design of Work, Third Edition, CRC Press, 2005.

[138] M. Swann, "Ergonomics of Touch Screens," Ergonomic Solutions International.

[139] E. Zidianakis, *Supporting Young Children in Ambient Intelligence Environments,* Heraklion, 2015.

[140] D. Michel, K. Papoutsakis and A. Argyros, ""Gesture recognition for the perceptual support of assistive robots"," in *International Symposium on Visual Computing (ISVC 2014)*, Las Vegas, Nevada, USA, 2014.

[141] D. Kosmopoulos, K. Papoutsakis and A. Argyros, "Segmentation and classification of actions in the context of unmodeled actions," in *British Machine Vision Conference (BMVC 2014)*, Nottingham, UK, 2014.

[142] F. Paterno, "User Interface Design Adaptation," in *The Encyclopedia of Human-Computer Interaction, 2nd Ed.*, 2013.

[143] A. S. C. Savidis, "Unified User Interface Design: Designing Universally Accessible Interactions," *International Journal of Interacting with Computers,* 2004.

[144] S. C. a. R. B. Mike Jackson, "Software Evaluation: Tutorial-based Assessment," software Sustainability Institute, Edimburgh, United Kingdom, 2011.

[145] R. J. Lewis, "IBM Computer Usability Satisfaction Questionaires: Psychometric Evaluation and Instructions for Use," *International Journal of Human-Computer Interaction,* vol. 7, no. 1, pp. 57-78, 1995.

[146] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of ACM INTERCHI'93 Conference*, Amsterdam, The Netherlands, 1993.

[147] C. Karat, R. L. Campbell and T. Fiegel, "Comparison of empirical testing and walkthrough methods in user interface evaluation.," in *Proceedings ACM CHI'92 Conference*, Monterey, CA, May 3-7, 1992.

[148] C. S. A. &. A. D. Stephanidis, "Tools for User Interfaces for all. In I.Placencia-Porrero & R. Puig de la Bellacasa (Eds.), The European context for Assistive Technology," in *Proceedings of 2nd TIDE Congress, IOS Press, pp. 167-170.*, Brussels, Belgium, Amsterdam, 1995.

[149] *Stephanidis, C. (2001) The concept of Unified User Interfaces. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools. Mahwah, NJ: Lawrence Erlbaum Associates, pp. 371-388 (ISBN 0-8058-2967-9).,* Mahwah, NJ: Lawrence Erlbaum Associates, 2001.

[150] A. S. C. Savidis, "Unified User Interface Development: Software Engineering of Universally Accessible Interactions," *Universal Access in the Information Society,* 2004.

[151] D. S. A. a. S. C. Akoumianakis, "Encapsulating Intelligent Interactive Behaviour in Unified User Interface Artefacts," *International Journal of Interacting with Computers, special issue on "The Reality of Intelligent Interface Technology",* pp. 383-408, 2000.

[152] A. S. C. a. E. P. Savidis, "Abstract Task Definition and Incremental Polymorphic Physical Instantiation: The Unified Interface Design Method. In G. Salvendy, M.J. Smith & R.J. Koubek (Eds.), Design of Computing Systems: Cognitive Considerations," in *Proceedings of the 7th International Conference on Human-Computer Interaction (HCI International '97)*, San Francisco, USA, 1997.

[153] C. A. D. S. M. a. P. A. Stephanidis, "Universal accessibility in HCI: Process-oriented design guidelines and tool requirements. In C. Stephanidis & A. Waern (Eds.)," in *Proceedings of the 4th ERCIM Workshop on "User Interfaces for All"*, Stockholm, Sweden, 19-21 October (15 pages), 1998.

[154] C. S. A. a. A. D. Stephanidis, *Tutorial on "Universally accessible UIs: The unified user interface development". Tutorial in the ACM Conference on Human Factors in Computing Systems (CHI 2001),* Seatle, Washington, 31 March - 5 April, 2001.

[155] C. E. P. Stephanidis, "Connecting to the Information Society: a European Perspective," *Technology and Disability Journal,* pp. 21-24, 1999.

[156] A. S. C. Savidis, The Unified User Interface Software Architecture. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools (pp. 389-415), Mahwah, NJ: Lawrence Erlbaum Associates, 2001.

[157] G. R. a. M. Arbib, "Language within our grasp.," *Trends in Neurosciences,* vol. 21, no. 5, p. 188–194, 1998.

[158] R. W. C. A. S. M. N. H. K. F. H. L. a. J. S. Y. Sakagami, "The intelligent ASIMO: system overview and integration.," in *Conference on Intelligent Robots and System*, Switzerland, 2002.

[159] R. Edlin-White, S. Cobb, M. D'Cruz, A. Floyde, S. Lewthwaite and J. Riedel, "Accessibility for Older Users through Adaptive Interfaces: Opportunities, Challenges and Achievements," in *J. Jacko (ed.) Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments*, Orlando, Florida, 2011.

[160] B. J. and S. M., "Psychologiczne aspekty starzenia się i starości. In: Fizjologia starzenia się. Profilaktyka i rehabilitacja. A. Marchewka, Z. Dąbrowski, J.A. Żołądź. Warszawa: Wydawnictwo Naukowe PWN, 2012, pp.48-59.".

Appendix I – The Alarm Clock Application: ACTA script

```
when(IsShowing == true)
{
    ActivateTimer();
}

when(IsShowing == false)
{
    DeactivateTimer();
}

when(SpeechRecognized == "GLOBALASKEDFORTIME")
{
    CheckIfInsideTimeSelection();
    Reason = "GLOBALASKEDFORTIME";
    NextState = "ClockScreen";
}

when(SpeechRecognized == "GLOBALASKEDFORALARMS")
{
    CheckIfInsideTimeSelection();
    Reason = "GLOBALASKEDFORALARMS";
    NextState = "AlarmsScreen";
}

when(SpeechRecognized == "GLOBALSETUPALARM")
{
    CheckIfInsideTimeSelection();
    NextState = "AddNewAlarm";
}

State "ClockScreen"
{
    ShowScreen("ClockScreen");
    when(Reason == "GLOBALASKEDFORTIME")
    {
        SpeakTime();
        Reason = "";
    }
    when(ButtonPressed == "Alarms")
    {
        NextState = "AlarmsScreen";
    }
}
```

```
State "AlarmsScreen"
{
    ShowScreen("AlarmsScreen");
    when(Reason == "GLOBALASKEDFORALARMS")
    {
        CheckAndSpeakAccordingly();
        Reason = "";
    }
    when(ButtonPressed == "Add")
    {
        NextState = "AddNewAlarm";
    }
    when(ButtonPressed == "Delete")
    {
        NextState = "DeleteAlarm";
    }
    when(ButtonPressed == "GoBack")
    {
        NextState = "ClockScreen";
    }
}

State "AddNewAlarm"
{
    ShowScreen("AddNewAlarm");
    when(ButtonPressed == "TIME_SELECTION")
    {
        NextState = InitiateTimeSelection();
    }
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        AddNewAlarm();
        NextState = "AlarmAdded";
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "AlarmsScreen";
    }
}

State "DeleteAlarm"
{
    ShowScreen("DeleteAlarm");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        DeleteAlarm();
        NextState = "AlarmDeleted";
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "AlarmsScreen";
    }
}
```

```
State "AlarmAdded"
{
    ShowScreen("AlarmAdded");
    when(Time == 4)
    {
        NextState = "AlarmsScreen";
    }
}

State "AlarmDeleted"
{
    ShowScreen("AlarmDeleted");
    when(Time == 4)
    {
        SelectedAlarm = null;
        NextState = "AlarmsScreen";
    }
}

State "AlarmElapsed"
{
    ShowScreen("AlarmElapsed");
    ReturnState = PreviousState;
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        DeleteAlarm();
        NextState = ReturnState;
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "AlarmSnoozed";
    }
}

State "AlarmSnoozed"
{
    ShowScreen("AlarmSnoozed");
    SnoozeAlarm();
    when(Time == 5)
    {
        NextState = ReturnState;
    }
}
```

Appendix II – The Alarm Clock Application: SRGS Grammar

```
<?xml version="1.0"?>
<grammar xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:xsd="http://www.w3.org/2001/XMLSchema" root="main" xml:lang="en-US"
tag-format="semantics-ms/1.0" version="1.0"
xmlns="http://www.w3.org/2001/06/grammar">
    <rule id="main">
        <one-of>
            <item>
                <ruleref uri="#time" />
                <tag>$.Result="GLOBAL_ASKED_FOR_TIME";</tag>
            </item>
            <item>
                <ruleref uri="#alarms" />
                <tag>$.Result = "GLOBAL_ASKED_FOR_ALARMS"; $.ShouldRespond
= $alarms.ShouldRespond;</tag>
            </item>
            <item>
                <ruleref uri="#alarmsetup" />
                <tag>$.Result = "GLOBAL_ALARM_SETUP"; $.Hours =
$alarmsetup.Hours; $.Mins = $alarmsetup.Mins; $.AMPM = $alarmsetup.AMPM;
</tag>
            </item>
        </one-of>
    </rule>

    <rule id="time">
        <one-of>
            <item>What time is it?</item>
            <item>What's the time?</item>
            <item>
                <item repeat="0-1">Could you please tell me </item>
                <one-of>
                    <item>what's the time?</item>
                    <item>what time it is?</item>
                    <item>the time?</item>
                </one-of>
            </item>

        </one-of>
    </rule>

    <rule id="alarms">
        <one-of>
            <item>
                What's on schedule for today?<tag>$.ShouldRespond =
1;</tag>
            </item>
            <item>
                What are my alarms?<tag>$.ShouldRespond = 1;</tag>
            </item>
```

```
            <item>
                Show me my alarms<tag>$.ShouldRespond = 0;</tag>
            </item>
        </one-of>
    </rule>
    <rule id="alarmsetup">
        <item>
            Set up an alarm for
        </item>
        <item>
            <ruleref uri="#fulltime" />
            <tag>$.Hours = $fulltime.Hours; $.Mins = $fulltime.Mins; $.AMPM
= $fulltime.AMPM;</tag>
        </item>
    </rule>

    <rule id="fulltime">
        <item>
            <one-of>
                <item>
                    <ruleref uri="#numbers" />
                    <tag> $.Hours = $numbers.Number; </tag>
                </item>
                <item>
                    Ten<tag>$.Hours = 10;</tag>
                </item>
                <item>
                    Eleven<tag>$.Hours = 11;</tag>
                </item>
                <item>
                    Twelve<tag>$.Hours = 12;</tag>
                </item>
            </one-of>
        </item>
        <item>
            <ruleref uri="#minutes" />
            <tag>$.Mins = $minutes.Mins;</tag>
        </item>
        <item>
            <ruleref uri="#ampm" />
            <tag>$.AMPM = $ampm.AMPM;</tag>
        </item>
    </rule>

    <rule id="ampm">
        <one-of>
            <item>
                A.M.<tag>$.AMPM = "AM";</tag>
            </item>
            <item>
                P.M.<tag>$.AMPM = "PM";</tag>
            </item>
        </one-of>
    </rule>
```

```
<rule id="minutes">
    <one-of>
        <item>
            <item repeat="0-1">
                zero
            </item>
            <ruleref uri="#numbers" />
            <tag>$.Mins = $numbers.Number;</tag>
        </item>
        <item>
            <ruleref uri="#minuteteens" />
            <tag>$.Mins = $minuteteens.Number;</tag>
        </item>
        <item>
            <item>
                <ruleref uri="#minutetens" />
                <tag>$.Mins = $minutetens.Number;</tag>
            </item>
            <item>
                <ruleref uri="#numbers" />
                <tag>$.Mins = $.Mins * 10 + $numbers.Number;</tag>
            </item>
        </item>
    </one-of>
</rule>

<rule id="numbers">
    <one-of>
        <item>
            One<tag>$.Number = 1;</tag>
        </item>
        <item>
            Two<tag>$.Number = 2;</tag>
        </item>
        <item>
            Three<tag>$.Number=3;</tag>
        </item>
        <item>
            Four<tag>$.Number=4;</tag>
        </item>
        <item>
            Five<tag>$.Number=5;</tag>
        </item>
        <item>
            Six<tag>$.Number=6;</tag>
        </item>
        <item>
            Seven<tag>$.Number=7;</tag>
        </item>
        <item>
            Eight<tag>$.Number=8;</tag>
        </item>
        <item>
            Nine<tag>$.Number=9;</tag>
        </item>
    </one-of>
</rule>
```

```
    <rule id="minuteteens">
        <one-of>
            <item>
                Ten<tag>$.Number=10;</tag>
            </item>
            <item>
                Eleven<tag>$.Number=11;</tag>
            </item>
            <item>
                Twelve<tag>$.Number=12;</tag>
            </item>
            <item>
                Thirteen<tag>$.Number=13;</tag>
            </item>
            <item>
                Fourteen<tag>$.Number=14;</tag>
            </item>
            <item>
                Fifteen<tag>$.Number=15;</tag>
            </item>
            <item>
                Sixteen<tag>$.Number=16;</tag>
            </item>
            <item>
                Seventeen<tag>$.Number=17;</tag>
            </item>
            <item>
                Eighteen<tag>$.Number=18;</tag>
            </item>
            <item>
                Nineteen<tag>$.Number=19;</tag>
            </item>
        </one-of>
    </rule>


    <rule id="minutetens">
        <one-of>
            <item>
                Twenty<tag>$.Number=2;</tag>
            </item>
            <item>
                Thirty<tag>$.Number=3;</tag>
            </item>
            <item>
                Fourty<tag>$.Number=4;</tag>
            </item>
            <item>
                Fifty<tag>$.Number=5;</tag>
            </item>
        </one-of>
    </rule>
</grammar>
```

Appendix III – The Main Menu Application: ACTA script

```
when(SpeechRecognized == "SHOW_MENU")
{
    NextState = "MainMenu";
}

State "MainMenu"
{
    ShowScreen("Menu");
    when(
       (ButtonPressed == "NEXT" || SpeechRecognized == "NEXT_PAGE") &&
       CanGoNext
    )
    {
        CurrentScreen = CurrentScreen + 1;
        ShowScreen("Menu");
    }
    when(
       (ButtonPressed == "PREV" || SpeechRecognized == "PREVIOUS_PAGE") &&
       CanGoBack
    )
    {
        CurrentScreen--;
        ShowScreen("Menu");
    }
}
```

Appendix IV – The Time Selection Process: ACTA script

```
State "TimeSelectionNov1Hours"
{
    ShowScreen("TimeSelectionNov1Hours");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        NextState = "TimeSelectionNov2Mins";
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        TimeSelectionCompleted();
        NextState = AbortState;
    }
}

State "TimeSelectionNov2Mins"
{
    ShowScreen("TimeSelectionNov2Mins");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        NextState = "TimeSelectionNov3AMPM";
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "TimeSelectionNov1Hours";
    }
}

State "TimeSelectionNov3AMPM"
{
    ShowScreen("TimeSelectionNov3AMPM");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        TimeSelectionCompleted();
        NextState = ReturnState;
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "TimeSelectionNov2Mins";
    }
}

State "TimeSelectionAvg1Hours"
{
    ShowScreen("TimeSelectionAvg1Hours");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        NextState = "TimeSelectionAvg2Mins";
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
```

```
    {
        TimeSelectionCompleted();
        NextState = AbortState;
    }
}
```

```
State "TimeSelectionAvg2Mins"
{
    ShowScreen("TimeSelectionAvg2Mins");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        NextState = "TimeSelectionAvg3AMPM";
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "TimeSelectionAvg1Hours";
    }
}

State "TimeSelectionAvg3AMPM"
{
    ShowScreen("TimeSelectionAvg3AMPM");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        TimeSelectionCompleted();
        NextState = ReturnState;
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        NextState = "TimeSelectionAvg2Mins";
    }
}

State "TimeSelectionExpAllInOne"
{
    ShowScreen("TimeSelectionExpAllInOne");
    when(ButtonPressed == "BUTTON_POSITIVE")
    {
        TimeSelectionCompleted();
        NextState = ReturnState;
    }
    when(ButtonPressed == "BUTTON_NEGATIVE")
    {
        TimeSelectionCompleted();
        NextState = AbortState;
    }
}
```

Appendix V – A sample from the developed adaptive style hierarchies

```
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/interactivedesigner/2006"
                xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                xmlns:sys="clr-namespace:System;assembly=mscorlib"
                mc:Ignorable="d"
                xmlns:awe="clr-
namespace:CustomCodeUtils.AwesomeAttachedProperties;assembly=CustomCodeUtils"
                xmlns:bhv="clr-namespace:CustomCodeUtils;assembly=CustomCodeUtils"
                xmlns:conv="clr-
namespace:CustomCodeUtils.Converters;assembly=CustomCodeUtils"
>

<!-- THESE ARE OVERRIDDEN AT RUNTIME -->
<sys:Double x:Key="ButtonWidth">192.0</sys:Double>
<sys:Double x:Key="ButtonHeight">128.0</sys:Double>

<sys:Double x:Key="HalfButtonWidth">96.0</sys:Double>
<sys:Double x:Key="HalfButtonHeight">64.0</sys:Double>

<sys:Double x:Key="KeyboardNormalButtonWidth">100.0</sys:Double>
<sys:Double x:Key="KeyboardNormalButtonHeight">84.0</sys:Double>
<sys:Double x:Key="KeyboardShiftButtonWidth">200.0</sys:Double>
<sys:Double x:Key="KeyboardGRIntonationButtonWidth">204.0</sys:Double>
<sys:Double x:Key="KeyboardCapsLockButtonWidth">150.0</sys:Double>
<sys:Double x:Key="KeyboardSpacebarButtonWidth">308.0</sys:Double>

<sys:Double x:Key="KeyboardWidth">988.0</sys:Double>
<sys:Double x:Key="KeyboardHeight">310.0</sys:Double>

<Thickness x:Key="KeyboardButtonMargin">10, 5</Thickness>

<sys:Double x:Key="TextBlockFontSize">32.0</sys:Double>

<SolidColorBrush Color="AliceBlue" x:Key="FrameworkUCsBackground"/>
<SolidColorBrush Color="LightSkyBlue" x:Key="FrameworkUCsBorderBrush"/>

<SolidColorBrush Color="Gray" x:Key="DisabledButtonForeground"/>
<SolidColorBrush Color="LightGray" x:Key="DisabledButtonBackground"/>
<SolidColorBrush Color="White" x:Key="ButtonForeground"/>
<SolidColorBrush Color="CornflowerBlue" x:Key="ButtonBackground"/>
<SolidColorBrush Color="DeepSkyBlue" x:Key="ButtonBackgroundIsMouseOver"/>
<SolidColorBrush Color="DarkBlue" x:Key="ButtonBackgroundIsPressed"/>

<SolidColorBrush Color="AliceBlue" x:Key="DarkerBg"/>
<SolidColorBrush Color="SkyBlue" x:Key="MainMenuStripeBackground"/>

<LinearGradientBrush x:Key="NormalBorderBrush" EndPoint="0,1" StartPoint="0,0">
```

```
    <GradientStop Color="DodgerBlue" Offset="0.0"/>
    <GradientStop Color="SkyBlue" Offset="0.5"/>
    <GradientStop Color="DodgerBlue" Offset="1.0"/>
</LinearGradientBrush>

<!-- /THESE ARE OVERRIDDEN AT RUNTIME -->

<!-- Brushes : These are used to define the color for background, foreground,
selection, enabled etc of all controls
If you want to change the color of a control you can just chnage the brush; if you
want to add a new shape or change arrangement then also edit the template -->

<!-- NormalBrush is used as the Background for SimpleButton, SimpleRepeatButton --
>
<LinearGradientBrush x:Key="NormalBrush" EndPoint="0,1" StartPoint="0,0">
    <GradientStop Color="#EEE" Offset="0.0"/>
    <GradientStop Color="#CCC" Offset="1.0"/>
</LinearGradientBrush>

<!-- LightBrush is used for content areas such as Menu, Tab Control background -->
<LinearGradientBrush x:Key="LightBrush" EndPoint="0,1" StartPoint="0,0">
    <GradientStop Color="#FFF" Offset="0.0"/>
    <GradientStop Color="#EEE" Offset="1.0"/>
</LinearGradientBrush>

<!-- MouseOverBrush is used for MouseOver in Button, Radio Button, CheckBox -->
<LinearGradientBrush x:Key="MouseOverBrush" EndPoint="0,1" StartPoint="0,0">
    <GradientStop Color="#FFF" Offset="0.0"/>
    <GradientStop Color="#AAA" Offset="1.0"/>
</LinearGradientBrush>

<!-- PressedBrush is used for Pressed in Button, Radio Button, CheckBox -->
<LinearGradientBrush x:Key="PressedBrush" EndPoint="0,1" StartPoint="0,0">
    <GradientStop Color="#BBB" Offset="0.0"/>
    <GradientStop Color="#EEE" Offset="0.1"/>
    <GradientStop Color="#EEE" Offset="0.9"/>
    <GradientStop Color="#FFF" Offset="1.0"/>
</LinearGradientBrush>
<LinearGradientBrush x:Key="PressedBorderBrush" EndPoint="0,1" StartPoint="0,0">
    <GradientStop Color="#444" Offset="0.0"/>
    <GradientStop Color="#888" Offset="1.0"/>
</LinearGradientBrush>

<!-- SelectedBackgroundBrush is used for the Selected item in ListBoxItem,
ComboBoxItem-->
<SolidColorBrush x:Key="SelectedBackgroundBrush" Color="#DDD"/>

<!-- Disabled Brushes are used for the Disabled look of each control -->
<SolidColorBrush x:Key="DisabledForegroundBrush" Color="#888"/>
<SolidColorBrush x:Key="DisabledBackgroundBrush" Color="#EEE"/>
<SolidColorBrush x:Key="DisabledBorderBrush" Color="#AAA"/>

<!-- Used for background of ScrollViewer, TreeView, ListBox, Expander, TextBox,
Tab Control -->
<SolidColorBrush x:Key="WindowBackgroundBrush" Color="#FFF"/>

<!-- DefaultedBorderBrush is used to show KeyBoardFocus -->
<LinearGradientBrush x:Key="DefaultedBorderBrush" EndPoint="0,1" StartPoint="0,0">
    <GradientStop Color="#777" Offset="0.0"/>
```

```
    <GradientStop Color="#000" Offset="1.0"/>
</LinearGradientBrush>

<SolidColorBrush x:Key="SolidBorderBrush" Color="#888"/>
<SolidColorBrush x:Key="LightBorderBrush" Color="#AAA"/>
<SolidColorBrush x:Key="LightColorBrush" Color="#DDD"/>

<!-- Used for Checkmark, Radio button, TreeViewItem, Expander ToggleButton glyphs
-->
<SolidColorBrush x:Key="GlyphBrush" Color="#444"/>


<!-- Style and Template pairs are used to define each control Part -->
<!-- The Style provides default values on the control; the Template gives the
elements for each control -->

<!-- SimpleButtonFocusVisual is used to show keyboard focus around a SimpleButton
control -->
<Style x:Key="SimpleButtonFocusVisual">
    <Setter Property="Control.Template">
        <Setter.Value>
            <ControlTemplate>
                <Border>
                    <Rectangle Margin="2" Stroke="#60000000" StrokeThickness="1"
StrokeDashArray="1 2"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>


<!-- Simple Button - This control sets brushes on each state. Note that these
brushes must be listed above since they are static resources -->
<Style x:Key="SimpleButton" BasedOn="{x:Null}">
    <Setter Property="Control.FocusVisualStyle" Value="{DynamicResource
SimpleButtonFocusVisual}"/>
    <Setter Property="Control.Background" Value="{DynamicResource NormalBrush}"/>
    <Setter Property="Control.BorderBrush" Value="{DynamicResource
NormalBorderBrush}"/>
    <Setter Property="Control.Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type ButtonBase}">

                <!-- We use Grid as a root because it is easy to add more elements
to customize the button -->
                <Grid x:Name="Grid">
                    <Border x:Name="Border"
                            Background="{TemplateBinding Background}"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}"
                            Padding="{TemplateBinding Padding}"
                            CornerRadius="5"/>

                    <!-- Content Presenter is where the text content etc is placed
by the control -->
                    <!-- The bindings are useful so that the control can be
parameterized without editing the template -->
```

```
                    <ContentPresenter HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
                                      Margin="{TemplateBinding Padding}"
                                      VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"
                                      RecognizesAccessKey="True" />
                </Grid>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- This BitmapEffect is used by the Simple MenuItem -->
<DropShadowBitmapEffect x:Key="PopupDropShadow" ShadowDepth="1.5"
Softness="0.15"/>

<conv:TextToVisibilityConverter x:Key="myTextToVisibilityConverter"/>

<BitmapImage x:Key="DefaultImage"
UriSource="pack://application:,,,/ApplicationsFramework;component/Images/genericic
on.png" />

<Style x:Key="TileButton" TargetType="{x:Type Button}" BasedOn="{StaticResource
SimpleButton}">
    <Setter Property="Background" Value="{DynamicResource ButtonBackground}" />
    <Setter Property="Foreground" Value="{DynamicResource ButtonForeground}"/>
    <Setter Property="Width" Value="{DynamicResource ButtonWidth}"/>
    <Setter Property="Height" Value="{DynamicResource ButtonHeight}"/>

    <Setter Property="FontSize" Value="{DynamicResource TextBlockFontSize}" />
    <Setter Property="Margin" Value="10"/>
    <Setter Property="ContentTemplate">
        <Setter.Value>
            <DataTemplate>
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition Height="*"/>
                        <RowDefinition Height="auto"/>
                    </Grid.RowDefinitions>
                    <Image Grid.Row="0" Grid.RowSpan="2"
                           VerticalAlignment="Bottom"
                           Margin="5"
                       Source="{Binding Path=(awe:ButtonEyeCandy.Image),
                                        RelativeSource={RelativeSource FindAncestor,
                                            AncestorType={x:Type Button}},
FallbackValue={StaticResource DefaultImage}}"

bhv:GrayOutOnDisabledImageBehavior.GrayOutOnDisabled="True"
                           IsEnabled="{Binding Path=IsEnabled,
                                               RelativeSource={RelativeSource
FindAncestor,
                                                   AncestorType={x:Type Button}},
FallbackValue=True}"  />
                    <Border Grid.Row="1"
                            Background="{DynamicResource
ButtonBackgroundIsPressed}"
                            Width="{Binding Path=ActualWidth,
                                            RelativeSource={RelativeSource
FindAncestor,
```

```
                                                AncestorType={x:Type Button}}}"
                                Height="{Binding ElementName=ContentText,
Path=ActualHeight}"
                                Opacity=".4"
                                Visibility="{Binding ElementName=ContentText,
Path=Text, Converter={StaticResource myTextToVisibilityConverter}}"/>
                            <TextBlock Grid.Row="1"
                                       x:Name="ContentText" Text="{TemplateBinding
Content}"
                                       VerticalAlignment="Bottom"
                                       HorizontalAlignment="Stretch"
                                       TextAlignment="Center"
                                       TextTrimming="CharacterEllipsis"
                                       TextWrapping="Wrap"
                                       Visibility="{Binding Path=Text,
Converter={StaticResource myTextToVisibilityConverter}}"/>
                        </Grid>
                    </DataTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="{DynamicResource
ButtonBackgroundIsMouseOver}"/>
                </Trigger>
                <Trigger Property="IsPressed" Value="True">
                    <Setter Property="Background" Value="{DynamicResource
ButtonBackgroundIsPressed}"/>
                </Trigger>
                <Trigger Property="IsEnabled" Value="False">
                    <Setter Property="Background" Value="{DynamicResource
DisabledButtonBackground}"/>
                    <Setter Property="Foreground" Value="{DynamicResource
DisabledButtonForeground}"/>
                </Trigger>
            </Style.Triggers>
        </Style>

        <Style x:Key="HalfTileButton" TargetType="{x:Type Button}"
        BasedOn="{StaticResource TileButton}">
            <Setter Property="Width" Value="{DynamicResource HalfButtonWidth}"/>
            <Setter Property="Height" Value="{DynamicResource HalfButtonHeight}"/>
        </Style>

        <Style x:Key="FrameworkUCs" TargetType="{x:Type UserControl}">
            <Setter Property="Background" Value="{DynamicResource
        FrameworkUCsBackground}"/>
            <Setter Property="BorderBrush" Value="{DynamicResource
        FrameworkUCsBorderBrush}"/>
            <Setter Property="BorderThickness" Value="1"/>
            <Setter Property="MaxWidth" Value="1024"/>
            <Setter Property="MaxHeight" Value="720"/>
            <Setter Property="Effect">
                <Setter.Value>
                    <DropShadowEffect Color="Gray" Opacity="0.2" />
                </Setter.Value>
            </Setter>
        </Style>
```

```
<Style x:Key="FrameworkKeyboardUC" TargetType="{x:Type UserControl}"
BasedOn="{StaticResource FrameworkUCs}">
    <Setter Property="MaxWidth" Value="1400"/>
    <Setter Property="MaxHeight" Value="720"/>
</Style>

<Style TargetType="{x:Type TextBlock}">
    <Setter Property="TextWrapping" Value="WrapWithOverflow"/>
    <Setter Property="TextAlignment" Value="Center"/>
    <Setter Property="FontSize" Value="{DynamicResource TextBlockFontSize}"/>
    <Setter Property="VerticalAlignment" Value="Center"/>
</Style>

<Style TargetType="{x:Type TextBox}">
    <Setter Property="FontSize" Value="{DynamicResource TextBlockFontSize}"/>
</Style>

<Style TargetType="{x:Type ComboBox}">
    <Setter Property="FontSize" Value="{DynamicResource TextBlockFontSize}"/>
</Style>

<Style x:Key="AlwaysSquareButton" BasedOn="{StaticResource SimpleButton}">
    <Setter Property="Control.MinWidth" Value="{Binding ActualHeight,
RelativeSource={RelativeSource Self}}" />
    <Setter Property="Control.MinHeight" Value="{Binding ActualWidth,
RelativeSource={RelativeSource Self}}" />
</Style>

<Style x:Key="SizelessCustomNormalButton" BasedOn="{StaticResource SimpleButton}">
    <Setter Property="Control.FontSize" Value="{DynamicResource
TextBlockFontSize}" />
    <Setter Property="Control.Margin" Value="10"/>

    <Setter Property="Control.Background" Value="{DynamicResource
ButtonBackground}" />
    <Setter Property="Control.Foreground" Value="{DynamicResource
ButtonForeground}"/>
    <Style.Triggers>
        <Trigger Property="Control.IsMouseOver" Value="True">
            <Setter Property="Control.Background" Value="{DynamicResource
ButtonBackgroundIsMouseOver}"/>
        </Trigger>
        <Trigger Property="Button.IsPressed" Value="True">
            <Setter Property="Control.Background" Value="{DynamicResource
ButtonBackgroundIsPressed}"/>
        </Trigger>
        <Trigger Property="ToggleButton.IsChecked" Value="true">
            <Setter Property="Control.Background" Value="{DynamicResource
ButtonBackgroundIsPressed}"/>
        </Trigger>
        <Trigger Property="Control.IsEnabled" Value="False">
            <Setter Property="Control.Background" Value="{DynamicResource
DisabledButtonBackground}"/>
            <Setter Property="Control.Foreground" Value="{DynamicResource
DisabledButtonForeground}"/>
        </Trigger>
    </Style.Triggers>

</Style>
```

```
<Style x:Key="CustomNormalButton" BasedOn="{StaticResource
SizelessCustomNormalButton}">
    <Setter Property="Control.Width" Value="{DynamicResource ButtonWidth}"/>
    <Setter Property="Control.Height" Value="{DynamicResource ButtonHeight}"/>
</Style>

<Style x:Key="SquareNormalButton" BasedOn="{StaticResource CustomNormalButton}">
    <Setter Property="Control.Width" Value="{DynamicResource SquareButtonDim}"/>
    <Setter Property="Control.Height" Value="{DynamicResource SquareButtonDim}"/>
</Style>

<Style x:Key="CustomNormalButton128" BasedOn="{StaticResource CustomNormalButton}"
>
    <Setter Property="Control.Width" Value="{DynamicResource ButtonHeight}"/>
</Style>

<!-- Keyboard Buttons -->
<Style x:Key="KeyboardNormalButton" BasedOn="{StaticResource
SizelessCustomNormalButton}" TargetType="{x:Type Button}">
    <Setter Property="Margin" Value="{DynamicResource KeyboardButtonMargin}"/>
</Style>
</ResourceDictionary>
```

Appendix VI – The IBM Computer Usability Satisfaction Questionnaires

The After-Scenario Questionnaire (ASQ)

The participant should fill-in this questionnaire after each scenario. For each of the statements below, circle the rating of your choice.

1. Overall, I am satisfied with the ease of completing the tasks in the scenario.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

2. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

3. Overall, I am satisfied with the support information (messages, documentation) when completing the tasks.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

The Computer System Usability Questionnaire (CSUQ)

The participant should fill-in this questionnaire once at the end of the evaluation (i.e. when all scenarios have been completed). For each of the statements below, circle the rating of your choice. The term "system" corresponds to the FIRMA framework.

1. Overall, I am satisfied with how easy it is to use this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

2. It is simple to use this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

3. I can effectively complete my work using this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

4. I am able to complete my work quickly using this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

5. I am able to efficiently complete my work using this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

6. I feel comfortable using this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

7. It was easy to learn to use this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

8. I believe I became productive quickly using this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

9. The system gives error messages that clearly tell me how to fix problems.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

10. Whenever I make a mistake using the system, I recover easily and quickly.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

11. The information (such as on-screen messages, feedback and other documentation) provided with this system is clear.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

12. It is easy to find the information I need.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

13. The information provided with the system is easy to understand.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

14. The information is effective in helping me complete my work.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

15. The organisation of information on the system screen is clear.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

16. The interface of this system is pleasant.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

17. I like using the interface of this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

18. This system has all the functions and capabilities I expect it to have.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

---

19. Overall, I am satisfied with this system.

Strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

Appendix VII – Scenario of Use (for the FIRMA framework) for Expert Evaluation Purposes

Objectives:

A) Create a new toy application and integrate it in the robot's menu. Complete the following steps in the order that they are provided. When required, you can refer to the sections of the provided tutorial as mentioned in the corresponding steps.

Steps:

1) Create a new Application (namely **BNApp**) that inherits from the RobotApp base class (tutorial section 1)
2) Create one dialogue (namely **BNScreen1**) inside the application that inherits from the RobotAppScreenBase base class (tutorial section 1)
3) Add a textbox to the BNScreen1 dialogue that invokes the virtual keyboard input dialogue when focused (tutorial section 2)
4) Create a second dialogue (namely **BNScreen2**) that derives from the binary decision dialogue, has a button which initiates the adaptive time selection task and updates its contents (tutorial section 3)
5) Add a button to the BNScreen1 with content "Next"
6) Add a button to the BNScreen2 with content "Back"
7) Change the buttons from steps 5 and 6 into TileButtons and add a background image (tutorial section 4)

Objectives:

B) Integrate multimodality, script application logic, localize the created application and add restrictions. When required, you can refer to the sections of the provided tutorial as mentioned in the corresponding steps.

Steps:

8) Create SRGS grammars for the two application dialogues (tutorial section 5)
9) Script the application's ACTA logic (tutorial section 6) that consolidates the touch, voice and gesture modalities to enable navigation back and forth between the two application's screens (BNScreen1 and BNScreen2).
10) Translate the new app in Greek (tutorial section 7)
11) Add a rule in the communication decision maker module that prohibits the user from initiating the time selection process (tutorial section 8)
12) Add an adaptation rule regarding the size of the displayed fonts and controls in respect to the distance between the robot and the user (tutorial section 9).