Project Identification	
Project number	AAL-2010-3-070
Duration	June 2011 – May 2014
Coordinator	Christopher Mayer
Coordinator Organisation	AIT Austrian Institute of Technology GmbH, Austria
Website	www.aaluis.eu

Ambient Assisted Living user interfaces

Document Identification	
Deliverable ID:	D-3.1 Report on user interface analysis
Release number/date	V1.0 19.12.2011
Checked and released by	Martin Morandell/AIT

Key Information from "Description of Work"	
Deliverable Description	The report presents the results of the User Interface Analysis from two perspectives: the end users perspective and the technical background.
Dissemination Level	PU=Public
Deliverable Type	R = Report
Original due date	Project Month 4 / 31 October 2011

Authorship& Reviewer Information	
Editor	Jan Bobeth / CURE
Partners contributing	Martin Morandell and Matthias Gira / AIT, Sascha Fagel / Zoobe
Reviewed by	Miroslav Bojic / PHIL

AALuis Consortium

AALuis (AAL-2010-3-070) is a project within the AAL Joint Programme Call 3. The consortium members are:

Partner 1	*AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GmbH (AIT, Project Coordinator, AT)*
Contact person:	Christopher Mayer
Email:	christopher.mayer@ait.ac.at
Partner 2:	*weTouch e.U. (weT, AT)*
Contact person:	Christian Schüler
Email:	christian.schueler@wetouch.at
Partner 3:	*Center for Usability Research & Engineering (CURE, AT)*
Contact person:	Jan Bobeth
Email:	bobeth@cure.at
Partner 4	*zoobe message entertainment GmbH (Zoobe, DE)*
Contact person:	Sascha Fagel
Email:	fagel@zoobe.com
Partner 5	*Verklizan BV (Verk, NL)*
Contact person:	Matti Groot
Email:	mgroot@verklizan.com
Partner 6	*ProSyst Software GmbH (PRO, DE)*
Contact person:	Kai Hackbarth
Email:	k.hackbarth@prosyst.com
Partner 7	*50plus GmbH (50plus, AT)*
Contact person:	Tanja Bosch
Email:	tanja.bosch@seniorenbund.com
Partner 8	*Hilfswerk Österreich (HWOe, AT)*
Contact person:	Walter Marschitz
Email:	walter.marschitz@hilfswerk.at
Partner 9	*Philips Consumer Lifestyle B.V. (PHIL, NL)*
Contact person:	Kees Tuinenbreijer
Email:	kees.tuinenbreijer@philips.com

Table of Contents

Table of Figures

List of Tables

Abbreviations

Abbrev.	*Description*
AAL	Ambient Assisted Living
AAL JP	Ambient Assisted Living Joint Programme
UI	User Interface
UIDL	User Interface Description Language
UX	User Experience

Executive Summary

The final goal of AALuis, to provide a flexible middleware layer to offer various services with a broad range of user interfaces in order to meet the needs of different people with disabilities and arising special needs, requires a flexible and extensible approach for user interface specification. User Interface Description Languages (UIDLs) have been introduced by research and industry to provide this kind of flexibility. In this deliverable the most promising UIDLs were analysed with respect to AALuis criteria with the result that MARIA XML, UsiXML and DISL are ranked best. The accordant criteria comprise level of abstraction, adaptability, openness and current status of development.

The opportunities of the AALuis middleware shall not only be demonstrated with existing services and user interfaces but also with the development of new and innovative approaches for interaction that are eligible and helpful for older people. For this reason existing knowledge and published ideas of interaction modalities for older people were explored, too. So the most promising approaches touch interaction, avatars and voice-based interactions were investigated in this deliverable. By this means we can stick to the collected guidelines and benefit from shared insights e.g. for touch interactions: to utilise eligible target and font designs, to avoid complex user interface elements and multiple input at one time as well as to focus at assuring good affordance and learnability.

As users of AALuis enabled devices shall have the choice out of a broad range of devices meeting their personal needs and contexts it is important to assure the best consistency possible. Users shall perceive the same service in different contexts and with different devices as all of a piece. For this reason this deliverable also introduces appropriate concepts like the three-dimensional model of interface consistency or the initial framework of cross-platform service user experience.

1 About this Document

1.1 Role of the deliverable

This deliverable presents the results of the user interface analysis for the AALuis project. We analysed relevant user interface issues from two perspectives: On the one hand side we investigated interaction modalities and consistency issues with respect to older people thus the end-users of AALuis, and one the other side we reviewed existing technical solutions in order to make use of existing specifications and therewith to avoid spending efforts on repetitive actions. As for the strict separation of services and its user interface abstraction levels are necessary the analysis of existing User Interface Description Languages (UIDL) was a key task for this deliverable.

The results of the analysis flow directly into the upcoming specification of the AALuis user interface layer and the user interface design. For this reason we also included relevant guidelines for designing user interfaces for older people in order to assure the creation of effective, efficient and satisfying user interfaces right from the start and before the conduction of first usability tests.

1.2 Relationship to other AALuis deliverables

The deliverable is related to the following AALuis deliverables:

Deliverable:	*Relation*
D3.2	The results of the analysis flow directly into the upcoming specification phase and thus have a direct influence on the description of the user interface specification.
D2.1	Parallel to the user interface analysis the analysis of middleware took place. The middleware will be the connecting element between user interface and provided service and thus the technical solutions have to stick together.
D4.1	Apart from user interface and middleware the existing services have been analysed. For providing the services to end users appropriate user interfaces have to be rendered and should therefore be able to fulfil their requirements.

2 Interaction Modalities for Older People

The AALuis platform will allow connecting a broad range of services with any kind of user interface. This has the advantage that every person gets access to every connected service by using their preferred interaction modalities and devices. As the development of new user interfaces for any group of users with several impairments and needs is far beyond the scope of this project we analyse those interactions modalities that are applicable for the defined target groups of the AALuis project.

2.1 Touch interaction: advantages and risks

Various age-related disabilities affect the way elderly can interact with computer systems. Chaparro et al. (1999) [4] and Wood et al. (2005) [53] have shown that the handling of computer with keyboard and mouse can cause problems with older people. Touch-interaction is commonly seen as an easier approach to engage older adults in interacting with computers, since it is a more direct interaction as with a computer mouse, without the need for an advanced mental model. The most obvious advantage of touch-screens is according to Greenstein and Arnaud (1988) [8] that the input device is also the output device. Besides systems using touch screens are not regarded as computers when using known metaphors (Vastenburg et al. [50]). In addition there is no need for special motor skills (Wood et al., 2005) [53]. However, designing touch interaction needs special attention, especially if it is supposed to be used by elderly people.

2.1.1 General (not age-related) insights from research

2.1.1.1 Evaluation of Touch strategies:

Potter et al. [30] evaluated three strategies for single touch input:

Land-On (L)	*First-Contact (F)*	*Take-off (T)*
Only first contact is used Selection only, if first contact hits a target	All position data is used until first contact with a target Selection upon first contact with a target	All position data is used until release Selection if target is in contact during release

Table 1: Strategies for single Touch input
There are three interaction modalities for single touch input: land on, First contact and Take off

They found the following:

Performance – Time

(F) showed the best results (16.93 sec) and was significantly faster than (T) (20.92 sec), while (L) did not differ significantly (17.73 sec).

Performance – Errors

(T) showed significantly fewer errors (mean=2.25) than (L) (mean=5.08) and (F) (mean=4.08).

Subjective Evaluation

(T) was significantly rated higher (mean=6.75) than (L) (mean=5.63) and (F) (mean=5.96).

2.1.1.2 Evaluation of Thumb Input for Touch Screens:

Perry and Hourcade (2008) [35] evaluated one handed thumb input tapping devices.

Hand used

- The movement time was 100ms higher when the non-preferred hand was used
- The accuracy was 5% lower when the non-preferred hand was used

Walking vs. Standing

- Walking vs. standing has no significant effect on moving time

Target Position

- The target position has on both hands a significant effect on movement time (the more central the faster) and accuracy (see Figure 1)

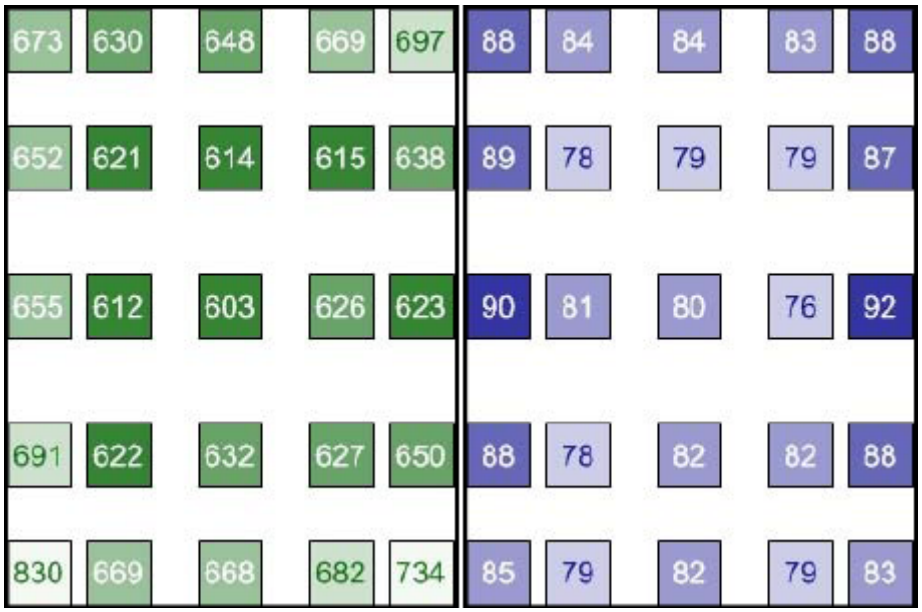

Figure 1: Movement Times
The left image shows the mean movement times for both hands (in ms), while the image to the right shows the mean accuracy rates for both hands [35].

Target Size

- The target size has on both hands a significant effect on accuracy

The study from Park et al. (2008) [34] showed that the button size has a significant influence on the number of errors, the success rate and the pressing convenience, meaning the larger the key, the lower the errors and the higher the success rate and pressing convenience. The results of a study by Parhi et al. (2006) [33]are that buttons should be approximately 9.2mm for a mobile device. At this size, the targets are “as small as possible without decreasing performance and user preference“. Also have look at the correspondent guidelines in chapter 2.1.3.

2.1.2 Age-related insights

2.1.2.1 Experiences with touch screens:

Touch screens that work for younger adults can cause problems for older adults [9]. Table 2 presents the results to the question whether the test persons have problems to use a touch terminal.

	Alter	1	2	3
Männer	20-29	100,0	0,0	0,0
	50-59	90,5	7,1	2,4
	60-69	85,2	6,6	8,2
Frauen	20-29	96,0	4,0	0,0
	50-59	88,0	4,0	8,0
	60-69	78,3	15,2	6,5

Table 2: Problems in using a Touch Terminal
Answers to the question whether test persons have problems to use a touch terminal:
(1) No and few problems (2) Avoidance of usage (3) problems

Screen size

For touch gestures being performed on larger screens deviation got larger but no relation with age could be observed [46].

Time & accuracy

According to Stößel (2009) [46], older users are „about 1.3 times slower […] than younger users“ but concerning accuracy of gesture execution no significant influence of age was found.

Apted et al. [1] examined how elderly people interact with a multi-user tabletop sharing system using single and multi-touch gestures. The results show that elderly users need nearly twice as long as younger users for learning and understanding the systems interaction paradigms but catch up while using the system afterwards.

Interaction and Navigation

Numerous studies developed and validated guidelines regarding the layout of information and interaction elements and the navigation through the system. Focusing the design of web sites, Kurniawan and Zaphiris (2005) [18] have developed a set of guidelines that take into account elderly users. According to this study it’s important to avoid deep hierarchy and to group information into meaningful categories. Complex interaction elements as pull down menus and scroll bars, as well as elements that have to be double-tapped, should be avoided. Furthermore, they recommend the system to avoid more than one open window, to align all interface elements in a horizontal and vertical grid and to concentrate the information in the center of the display. Important information should be highlighted.

In addition to these guidelines that were proposed with regard to conventional screens, some studies analyzed the special requirements of touch screen systems. As Yang (2008) [54] states, complex control techniques should be avoided – elderly people may find it hard to perform sliding and rotating touch screen gestures. The use of familiar gestures, i.e. gestures that correspond to the real world, reduces the error rate compared to standard touch-interaction (Guerreiro et al. 2008 [10], Roudaut 2009 [39]). Concerning the arrangement of information and interaction elements, headlines, as the major information,

should be displayed at the top and buttons on the bottom of the interface, so the input-hand would not hide the screen (Lorenz et al., 2007) [22]. Furthermore it should be considered, that the screen may be large, so users are unable to reach the whole screen easily (Apted et al. 2006) [1].

Information Input

Maguire (1999) [23] addressed information input on touch-screen devices specifically for elderly: Information input should be as simple as possible, only one input at a time. A sequence of prompts is preferred compared to a form-filling style of input. Furthermore it is important, to avoid requiring long textual inputs to the system and to clearly highlight the input position or focus on the screen. The virtual keyboard should provide a “Backspace” button to correct entered text and/or a “Clear” button to clear the whole input.

Target Design

The design of interactive targets, such as buttons or menus provides challenges first due to the very nature of the human fingers, the latter due to reduced motor skills. Both demand larger targets (Jin et al., 2007 [16]; Yang, 2008 [54]), but there are several other issues that need to be considered. While target size matters, bigger spacing between targets apparently does not improve usability (Schedlbauer, 2007 [42]; Sun 2007 [47]). Several studies have shown that elderly users demand clear and meaningful target captures (Kurniawan and Zaphiris, 2005 [18]; Lorenz et al., 2007 [22]) and, furthermore, the interface should allow to easily resize the target elements (Apted et al., 2006 [1]).

Use of Colour and Graphics

When using interactive systems elderly people might have difficulties with the use of colors, graphics, icons and background. To address these aspects many guidelines have been developed. Considering elderly people using web-based interfaces, Kurniawan and Zaphiris (2005) [18] found that they prefer the use of graphics if they belong to the content and are not just used for decoration. Animated elements tend to confuse older users and should therefore be avoided in general while animated avatars are considered helpful. Icons should be simple and meaningful (Kurniawan and Zaphiris, 2005) [18], need to be large enough to be identifiable by people with reduced eyesight (Bhachu et al., 2008) [3] and have to provide labels (Maguire, 1999) [23].

Regarding the use of color, for elderly people it is even more important than for younger users that the screen provides a high contrast between foreground and background (Kurniawan and Zaphiris, 2005 [18]; Maguire, 1999 [23]; Young, 2008 [54]). Therefore, colors should be used conservatively. The number of colors should be kept within reasonable limits (4 or 5) (Kurniawan and Zaphiris 2005 [18], Maguire 1999 [23]). Visually impaired users prefer color-neutral layouts (Lorenz et al., 2007) [22]. Combinations of blue/green, red/green and red/blue tones should be avoided (Kurniawan and Zaphiris, 2005 [18]; Maguire, 1999 [23]). Colors can be helpful to structure the display, group categories of data and to help identifying target elements (labels, entry fields, prompts etc.) (Maguire, 1999) [23]. Regarding touch screens, touch areas or screen buttons should be easily distinguishable from other graphics (Maguire 1999) [23].

Text Design

Regarding the special needs of visually impaired people, the layout and formatting of the text is an important factor regarding the interface design for elderly users. With regard to this, lessons learned from conventional screens are, to the greatest extent, also valid for touch screen interfaces. Small font sizes (< 12 pt.) and serif font types should be avoided (Yang, 2008 [54]; Lorenz et al., 2007 [22]; Kurniawan and Zaphiris, 2005 [18]). Text should

be left justified, have clear headings and the lines should be short in length (Kurniawan and Zaphiris, 2005 [18]; Maguire 1999 [23]). Enough spacing between the lines eases reading longer text segments (Kurniawan and Zaphiris 2005) [18]. Furthermore different fonts may be used to help distinguish between system messages and user entry (Maguire 1999) [18]. Also have look at the correspondent guidelines in chapter 2.1.3.

User Feedback

Touch interfaces mostly offer only a visual feedback when a button is pressed. This visual feedback is further interfered by the user's fingers, thumb or hand [34]. Multimedia interfaces improve the memorization of information. Only 10% of visual and 20% of audio feedback is remembered, while 50% of combined visual and auditory feedback stays in memory, and even 80% of hearing, seeing and acting together (Yang, 2008) [54]. This way, multimedia-based feedback (visual, aural) or speech output can provide a communicative and easy environment (Tsai, 2009 [49]). Jacko et al. (2004) [16] found that both novice and experienced older users can benefit from improved feedback combinations more significantly than the younger user groups. The results suggested that auditory feedback caused the greatest improvement in performance error reduction. However, audio feedback for lengthy inputs can be annoying to the user (Maguire, 1999 [23]).

If the system response to user input takes more than 2 or 3 seconds, users may start to feel that a fault has occurred (Maguire, 1999 [23]). For audio feedback it needs to be considered that, while frequency matters, according to Yang (2008) [54] modifying the speed of speech does not affect accessibility of the elderly. Lower sound frequencies are more appropriate for the older person (Hawthorn 2000 [12]).

Cognition and design of touch interfaces and gestures

To help people lacking knowledge about the standard window interaction metaphor and to avoid greater difficulties in learning new concepts and declining short term memory the user interface design should focus on learnability and memorability. Alternative solutions have been suggested, such as animated conversational agents and interfaces that rely on familiar aspects of manipulating physical objects and use metaphors (Apted et al., 2006 [1]; Stößel et al. 2009 [46], Yang 2008 [54]).

The number of interface elements should be minimized (Apted et al. [1], 2006; Lepicard and Vigouroux, 2010a [20]; Lepicard and Vigouroux, 2010b [21]). If new interface behaviors appear, new objects should be used to avoid clashes with the user's existing knowledge (Apted et al., 2006 [1]). Functionalities that require conceptual background knowledge should be avoided (Yang, 2008) [54].

In a multi touch study elderly users had problems with a specific gesture (the copy-gesture), because of not remembering it from the tutorial and of hesitating to interact with a second finger or hand (see Figure 2, out of Apted et al., 2006 [1]).

In general, multi-touch devices show potential to substantially alleviate interactions for elderly people. In particular, simple gestures (like crossing out, ticking off) seem to be easier to understand and to learn than traditional ways like pressing a certain button (Stößel 2009[46]).

Norman (2010) [30]identified a number of challenges that gestural interfaces must cope with, e.g. the lack of feedback and the lack of support for discovering specific functionalities. Since gestures are unconstrained, they are vulnerable to be performed in an ambiguous manner; in this case, constructive feedback is required to allow the person to learn the appropriate way to trigger a specific action.

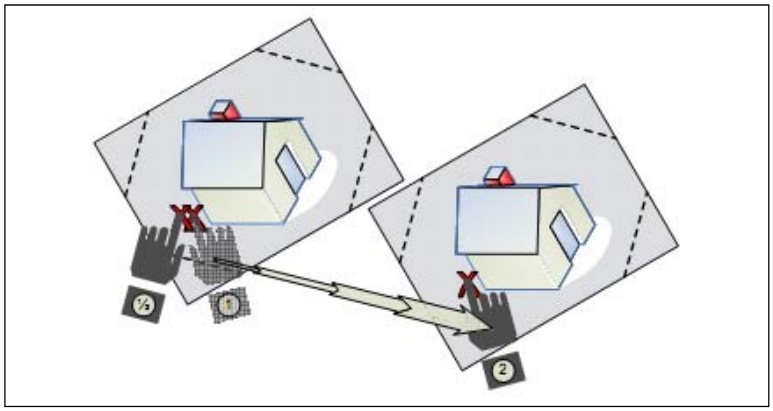

Figure 2: Copy gesture
The copy gesture: Left finger and picture remain stationary while the copy appears beneath the right finger, when it is moved away.

On the other hand, gestures may be easier to learn and remember when designed to reflect manipulations of real world physical objects. According to Guerreiro et al. (2008) [10] and Roudaut (2009) [39] familiar gestures reduce the error rate compared to standard touch interaction. However, the older adults' mental model of the world is significantly different than mental model of younger people due to advancement of technology in the years that passed. A 70 years old person has a different idea of how a telephone works than a 17 years old person, due to differences in telephones from the periods in which they grew up. This should be taken in account when replicating real world manipulations in interfaces for the elderly.

Wobbrock et al. (2009) [50] developed an approach for the design of multi-touch gestures that relies on eliciting gestures from non-technical users. The results of this study suggest that users don't care about the number of fingers used in the gesture, prefer one hand over two and are strongly influenced by the desktop paradigm. Furthermore, they found out that there is little agreement over what gesture would be appropriate for a specific command.

Interacting with gestures can be more entertaining and may encourage people to intuitively explore the interface. For this reason, multi-touch interfaces should be discoverable so that people can find out for themselves how they work (Saffer, 2008) [41].

2.1.2.2 Experiences with NFC-based touch interactions:

Häikiö et al. (2007) [11] developed a meal ordering system where a NFC-enhanced mobile phone was used as a user interface element so as to enable home-dwelling elderly people to choose their meals to be delivered by means of a home care service (see Figure 3). Most participants (average age of 76,6) had memory disorders of differing levels assessed with the Rava index [39] and only those elderly persons with lowered functional capabilities were accepted. In the training session all the participants adopted the touch interaction method easily regardless of their motor skills. Although the meal ordering became easier and faster (instead of calling) four out of nine participants preferred the older catering practice and did not think that the application would provide them with added value.

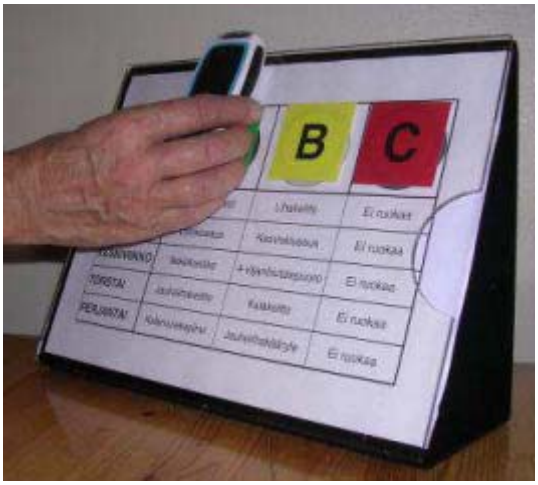

Figure 3: NFC Interface
For selecting their preferred meal users had to hold a NFC-enhanced mobile phone at the accordant letter
Experiences with Pen-based touch screens:

Moffatt and McGrenere (2007) [26] found out, that there are three main problems when using a stylus based interface. While all of them apply to older users and two to younger usurers as well:

1. Slipping: the target was selected, but before releasing (lifting the pen) the users unintentionally slip off (only older users)
2. Drifting: an adjacent menu is opened accidently by hovering over
3. Missing just below: instead of the target item, the one just below is selected via selecting the top edge

Further, Moffatt et al. (2008) [27] examined if drifting could be avoided by using new strategies like “Tap” or “Glide”. “Tap” requires an explicit click for changing menus while “glide” uses a distance threshold for controlling if a switch is desired or not.

2.1.2.3 Touch interaction vs. Pen interaction:

Iglesias et al. (2009) [15] compared touch interaction and RFID-based pen interaction for/with elderly users. The results show that touch interaction is “easy and comfortable for them” and that “they showed their preferences for the touch screen”. Regarding the time used for completing the given tasks, both methods are similar.

2.1.3 Guidelines

In this section we provide some guidelines for user interface design for touch screens with respect to older people that have been derived from various research investigations. Although some of the guidelines are especially derived for web design those presented here also apply for touch screens.

2.1.3.1 Web Design Guidelines for Elderly (Source: [18])

Target Design

- Provide larger targets
- There should be clear confirmation of target capture, which should be visible to older adults who should not be expected to detect small changes

- Older adult should not be expected to double click

Graphics

- Graphics should be relevant and not for decoration. No animation should be present
- Images should have alt tags
- Icons should be simple and meaningful

Navigation

- Extra and bolder navigation cues should be provided
- Clear navigation should be provided
- Provide location of the current page
- Avoid pull down menus
- Do not use a deep hierarchy and group information into meaningful categories

Browser Window Features

- Avoid scroll bars
- Provide only one open window e.g., pop-up/animated advertisements or multiple overlapping windows should be avoided

Content Layout Design

- Language should be simple and clear, and unambiguous
- Avoid irrelevant information on the screen
- Important information should be highlighted
- Information should be concentrated mainly in the centre
- Screen layout, navigation and terminology used should be simple, clear and consistent

User Cognitive Design

- Provide ample time to read information
- Reduce the demand on working memory by supporting recognition rather than recall and provide fewer choices to the user

Use of Colour and Background

- Colours should be used conservatively
- Blue and green tones should be avoided
- Background screens should not be pure white or change rapidly in brightness between screens. Also, a high contrast between the foreground and background should exist, for example, coloured text on coloured backgrounds should be avoided.
- Content should not all be in colour alone (colour here is denoted by all colours other than black and white)

Text Design

- Avoid moving text
- Text should be left justified and text lines should be short in length
- There should be spacing between the lines but not too much either
- Main body of the text should be in sentence case and not all capital letters
- Text should have clear large headings
- Use san serif type font i.e., Helvetica or Arial of at least 12 point size. Avoid other fancy font types.

User Feedback & Support

- Provide a site map
- An online help tutorial should be provided
- Support user control and freedom
- Error messages should be simple and easy to follow

2.1.3.2 Heuristics for Older Adults as Web Users (Source: [5])

Use conventional interaction elements.

- Does the site use standard treatments for links?
- Is link treatment the same from section to section within the site?

Make clickable items easy to target and hit.

- Are buttons large enough to easily see the image or text on them?
- Is the area around buttons clickable?
- Is there enough space between targets to prevent hitting multiple or incorrect targets?
- Do buttons and links enlarge when the rest of the text size is increased?

Minimize vertical scrolling; eliminate horizontal scrolling.

- Does the site work at the resolution at which the user would typically view the site without horizontal scrolling?
- Do pop-ups and secondary windows open wide and long enough to contain the content without the need for scrolling?
- For scrolling lists, for example, a list of all the states:
- Are checkboxes used rather than drop-down (a menu that drops down when requested and stays open without further action until the user closes it or chooses a menu item) or pull-down menus (a menu that is pulled down and that stays available as long as the user holds it open)?
- If not, are drop-down menus used rather than pull-down menus?

Let the user stay in control.

- Is there no rolling text that goes by automatically?

- Does the site use static menus (a click leads to another page) rather than “walking menus” (exposing a sub-menu on hovering the mouse over the label)?
- If there are walking menus, do they expand on a click (rather than a hover)?
- Are the sub-menus timed to stay open for at least5 seconds or until they’re clicked?

Provide clear feedback on actions.

- Are error pages descriptive, and did they provide a solution to the user?
- Are confirmation pages clear?

Provide feedback in other modes in addition to visual.

- Are captioning and/or meaningful alternative text provided for images, video, and animation?
- Does the site support haptics?

Clearly label content categories; assist recognition and retrieval rather than recall.

- Are labels descriptive enough to make it easy to accurately predict what the content will be under each topic category?
- Do labels and links start with different, distinct, and relevant key words?
- Are labels useful and understandable each on their own?
- Do labels reflect language that older adults are familiar with?

Implement the shallowest possible information hierarchy.

- Are important, frequently needed topics and tasks closer to the surface of the Web site?
- Are related topics and links grouped and labelled?
- Do labels and category names correspond to users’ tasks and goals?
- Do paths through the information architecture support user’s tasks and goals?
- Is the path for any given task a reasonable length (2–5 clicks)?
- Is the path clear of distractors and other obstacles to reaching task goals?
- Are there a few, helpful cross-referenced links that are related to the current task goal?
- Do redundant links have the same labels?

Make pages easy to skim or scan.

- Are pages clean looking and well organized (versus cluttered or busy)?
- Is there a clear visual “starting point” to the page?
- If pages are dense with content, is content grouped or otherwise clustered to show what is related?
- Is it easy to tell what is content and what is advertising?
- Do task-supporting keywords stand out?
- Are images relevant to, and supportive of, the text content?

- If there are videos or animated sequences, do they support specific goals or tasks?

Make elements on the page easy to read.

- Is the default type size 12-point or larger?
- Is the type size on pull-downs and drop-down menus the same size as the text content? Does it change when the user increases the type size?
- Are headings noticeably larger than body content (18- or 24-point)?
- Is sans serif type used for body content?
- Are headings set in a typeface that is easy to read?
- Are there visual cues to direct users' attention to important items that are in the left and right columns?

Visually group related topics.

- Is the amount of information—sparse, dense, or in between—appropriate for the audience and type of site?
- Are the most important and frequently used topics, features, and functions, close to the centre of the page rather than in the far left or right margins?
- Are task-related topics grouped together?
- Are frequently used topics, actions, and links „above the fold"?

Make sure text and background colours contrast.

- Are text and interaction elements a different colour from the background (not just a different hue)?
- Do the colours that are used together make information easy to see and find?
- Are clickable items highlighted differently from other non-clickable highlighted items?
- Are multiple types of highlighting minimized on each page?

Use adequate white space.

- Are there visual cues in the layout of the page that help users know there is more content "below the fold"?
- Are there at least 2 pixels of line space between clickable items?
- Is body text broken up with appropriate and obvious headings?

Make it easy to find things on the page quickly.

- Is the amount of text minimized; is only necessary information present?
- If there are introduction paragraphs, are they necessary?
- Are instructions and messages easy to recognize?
- Is there liberal use of headings, bulleted lists, and links to assist skimming?
- Do bulleted lists have the main points and important keywords at the beginning of each item?
- Do links have meaningful labels?

- Are buttons labelled clearly and unambiguously?
- Do button and link labels start with action words?

Focus the writing on audience and purpose.

- Is the content written in active voice, directed to “you”?
- Are sentences short, simple, and straightforward?
- Are paragraphs short?
- If humour is used, is it appropriate?
- Are headings, labels, and captions descriptive of associated content?
- Are conclusions and implications at the top of a body of text, with supporting content after? (inverted pyramid)

Use the users’ language; minimize jargon and technical terms.

- Does the site use words that older adults know?
- If there are technical words or jargon, are they appropriate for the level of domain expertise that the audience has?
- If there are new or technical terms, does the site help users learn what the terms mean?
- Are concepts and technical information (such as safety and effectiveness information about a prescription drugs) written in plain language?
- Are instructions written in plain language?
- Is the reading level appropriate for the capabilities of the audience and their literacy in the topic area? Is it easy to draw inferences and to understand the implications of text?

2.1.3.3 Heuristics for Tabletop Systems (Source: [1])

Design independently of table size Design for different tabletop sizes and allow flexible resizing of all interface elements.

Interfaces should not be constrained to a particular table size, and designers must consider that some interface elements may need to be regularly enlarged depending on the task or to be legible by all users (particularly those with restricted eyesight or the elderly).

Support reorientation

Allow all interface elements to be easily rotated to support users working at any position around the table, and consider users moving around the table while using it. People should be able to view and interact with the table at any position around it, and interface elements should be easy to rotate.

Minimise human reach

Consider that users may not be able to physically reach all interface elements. Elements must be moveable to all areas of the table. Interface elements should not be fixed, as that could constrain users’ positions at the table, and cause usability problems if some

elements are unreachable. Designers must also consider social expectations that prevent users from reaching in front of others.

Use large selection points

Design independently of table top input hardware, but support large input cursors (e.g. human fingers) where possible. To provide a natural interface with the restricted input available at table tops, interface elements should be usable through direct-touch interaction with large fingers or other input styli.

Manage interface clutter

Support quick removal or hiding of objects on the table top, while ensuring management of clutter by one user does not have unwanted side-effects on other users of the table. Clutter management is a significant problem to address in table top interface design, due to constraints on display and input technology, table size, and supporting multiple people working concurrently.

Use table space efficiently

Avoid modal behaviour that limits the utilisation of table space. Allow arbitrary groupings of interface elements for personal and group spaces. Modal behaviours, such as confirmation dialogs that take focus away from the rest of the table, constrain multi-user collaborative interaction. It is also important to support people forming personal and group spaces— a natural tendency when collaborating at a table top.

2.2 Avatars: an opportunity to involve older people

The acceptance of the applied assistive technology is the crucial factor if the user can derive a benefit from the applied technology or not. Beside the fact that the benefit of using new devices must be appreciable, in order to provide a motivation for its use [78], there can even be "a need for an emotional relationship" between the users and their Assistive Technology as described by Wu P. 2005 [84].

In addition the paradigm shift from "the computer used as a tool" to "the computer used as an assistant", the vision of ICT for everyone without any barriers that might cause a "digital divide" and the request to make technology friendly, polite and fun to use have caused a widespread use of avatars in many fields (e.g. internet pages, movies, computer games and kiosk situations).

2.2.1 What is an avatar?

An avatar can be seen as consisting of "body and mind" (compare to Spierling 200649[82]): A software agent builds human models like emotional states, models of cognition and knowledge. The software agent can be seen as the **mind of the avatar**.

These models are presented to the exterior using visually human models. This visualization can be seen as the **body of the avatar**

To use photo realistic talking heads can be a method to make a computer based system being perceived more personal by people with dementia. Like this the acceptance of such a system can be increased. This effect is especially fostered by using photos of known faces as basis to create talking heads.

At AIT studies have been carried out concerning the use of artificial talking heads as part of the user interfaces for assistive devices for elderly people – with and without dementia.

The studies were embedded into a master thesis [80] and in the project Avatars@Home (in cooperation with CURE) [81]. The results are summarized in the following chapters.

2.2.2 Perception of Avatars

Selection of faces for avatars:

When also known faces were offered, tests showed that only photos of familiar persons were chosen to be used as instruction giving talking heads. The combination of relatives and known formal caregivers seem to fit the wishes of the subjects. For further testing but also for the real setting of such a system, photos of more relatives should be available.

2.2.2.1 Audio Visual perception:

The visual impression of the talking head influenced the acoustic perception, leading to the impression that even the used synthetic voice was the original voice of the underlying person of the talking head. This was even the case by subjects being confronted with an avatar of their own children. Especially if voice and face were felt matching, the avatar based user interfaces are more preferred then text and voice interfaces only.

2.2.2.2 EyeCatcher Effect

When a new scene is displayed at the screen and the user is not aware of it, avatars can be used as a kind of eye catcher to get the user's attention and also to keep it on the screen where also the task message is displayed and spoken by the avatar.

Just "calling" the user, like "Hello Mum" or "Mrs. B..." is not enough. Like this the shown head helps to “localize” the origin of the perceived voice. „The most crucial point is that using photo realistic avatars can lead to interpersonal reactions - with positive and negative effects”. This strongly relates to the Uncanny Valley:

2.2.3 The Uncanny Valley

Masahori Mori describes the human psychological reaction to robotic design, which is also applicable to interactions with nearly any non-human entity [83]. The idea is that if a curve describes the emotional response against similarity to human appearance and movement, the curve is not a sure, steady upward trend. Instead there is a peak shortly before the inhumane entity reaches a completely human “look”. This peak is followed by a strong negative response before rebounding to a second peak where resemblance to humanity is complete.

In this uncanny valley between the two peaks, a person sees a figure or object that looks nearly human but enough off-kilter to seem eerie or disquieting [79]. To avoid falling in the “uncanny valley”, M. Mori suggests designers of robots or even developers of prosthetic hands to take the first peak as goal rather than the second. Even though the second peak is higher, there is a far greater risk of falling into the uncanny valley

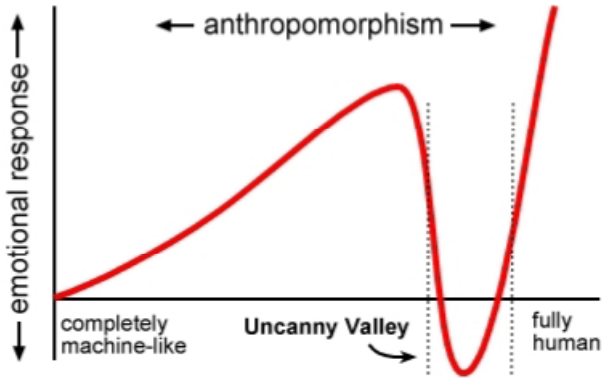

Figure 4: The uncanny Valley
This graph describes the uncanny valley. The more natural a character looks like, the more it is "liked" until a certain point, when it cannot be said, whether the character is human or not. This valley is combined with fear. When the character reaches a very high level of anthropomorphism, where it is realized as human, this likability rises again.

Virtual "humans" also raise this curve concerning the relation between their reached level of humanity and the viewers perceived feeling of familiarity. Two and three dimensional computer animated characters are still not looking human, especially in their movements.

Human beings are especially skilled in recognizing faces. This on one side allows recognizing a person that has not been met for a long period. On the other side, if a single part of a known face does not comply with the familiar appearance, this fact is detected immediately most of the time.

Till now, no information could be found about the uncanny valley of "virtual characters" and persons with cognitive impairments, in particular persons with dementia. Demented person soften have problems recognizing known faces, even of their spouses or their own children. If an avatar or a talking head is created by a photo of a well known person, how will a demented person perceive the (up to now still inescapable) unnatural appearance of the (known) human head especially during movements as talking [80]?

Some research can be found on dolls, artificial pets, etc. and the likeability by people with dementia [85][86].

2.3 The chances of Voice-based interaction

Enabling voice interaction reveals various chances. First of all, if the visual sense is handicapped the service system can provide linguistic information to the users by voice output. This is also the case for situations where the spatial configuration does not allow to place the display of the device that serves as user interface of the service into the field of view of the user - e.g. in the dark bedroom - , or if the user needs his/her visual attention elsewhere - e.g. while operating the oven for the preparation of meals.

In the other direction, i.e. voice input to the service system, voice can replace input modalities like typing a text or touching a visual representation of a user interface item (like pressing a button or selecting an icon). This again is especially useful if the user cannot actually see the user interface items like keyboard or button/icon due to visual impairment, or the user interface device is out of reach or the user needs his/her hands for other purposes than to handle the device, e.g. while moving a wheelchair.

Voice interaction is especially useful in the context of Ambient Assisted Living for elderly as the users of such systems and services often have special essential needs and at the same time reduced abilities to handle user interface devices.

3 User Interface Description Languages[1]

In order to reach the desired flexibility of user interfaces within the AALuis approach proprietary solutions are not feasible. For this reason the usage of user interface description languages (UIDL) is necessary. Instead of creating an UI for a specific platform the UI will be modelled in a more abstract UIDL format. This section analyses existing UIDLs with respect to the AALuis criteria.

3.1 Analysis criteria for UIDLs in AALuis

Level of abstraction

UIDLs must be sufficiently abstract to allow the creation of multimodal user interfaces, meaning user interfaces for different devices that use different interaction modes. For example, the user interaction with a PC with mouse and keyboard is totally different from that with a mobile phone with multi-touch-screen, or a car-radio featuring only voice interaction.

Requirements for a high level of abstraction are that no information about the use of specific user interface widgets is encoded, as well as that no concrete layout information is given since this information might only be used by graphical user interfaces on a certain type of display.

Adaptability

This criterion concerns the possibility to adapt user interfaces automatically based on different environmental settings. For the AALuis project, the following three characteristics are important:

- Accessibility: The user interface should automatically adapt to user preferences, based on a user's abilities and disabilities. This is important especially to users of AAL environments, because most of them use these systems to overcome physical disabilities.
- Use-case awareness: In different use cases, different UI devices are used. For example, for activating and deactivating a service, a mobile phone is mostly used because the user can carry it with him, while changing the basic setup of a service is carried out using a PC due to the more sophisticated input mechanisms.
- UIDLs should know about the capabilities of UI devices and automatically adapt user interfaces to provide different functionalities on different devices as well as present the user interface in the way that is supported best by each type of device.
- Context-awareness: Finally, it is desirable to automatically adapt the presentation of a user interface based on environmental influences, for example physical conditions such as the intensity of light around the UI device.

Openness

This criterion evaluates how much information about a UIDL is freely available, and how detailed this information is. In addition, it concerns licensing issues.

[1] The content of this section is based on the Master Thesis “User Interaction Description Languages to create accessible interfaces for elderly people“ by Andreas Kuntner

The aim of this criterion is to give an overview of the history and the current development status of a UIDL. It answers the following questions:

- When did the development of the first version start?
- How many versions have been published?
- Which is the latest version, and when has it been published?
- Is the language still further developed?
- Is a standardisation process of the language specification planned, or has it been accepted as national / international standard already?

It is also stated which type of organisation started the development of a UIDL, and if it has an industrial or a research background.

3.2 Analysis of UIDLs

Various researches have been conducted on the comparison of UIDLs. For this reason we refer in this section often to secondary literature.

3.2.1 ISO/IEC 24752 Universal Remote Control, Presentation Template Mark-up Language (PreT)

The standard titled “Information technology - User interfaces - Universal remote console” [56]defines the Universal Remote Console (URC) specification. The basic setup of a URC environment consists of one or multiple target devices that provide services the user can access through the use of one or multiple user interface devices. The concrete user interface is rendered individually for the UI device’s platform at runtime, based on the capabilities of the available services.

The capabilities of the target services are specified in target descriptions, their public interfaces to be used by user interface devices are specified in user interface sockets, and abstract user interfaces are specified for each target service in presentation templates. All definitions are made available through XML documents.

User interface sockets are *“an abstract concept that, when implemented, exposes the functionality and state of a target in a machine-interpretable manner”* [57]. A user interface socket contains all information that is internally present in a target service implementation and should be made accessible to the user. This includes (public) variables and functions as well as notifications.

A Presentation Template (PreT) as defined in the presentation template mark-up language maps elements of a user interface socket to interaction mechanisms. It contains abstract interactors that are used for either presenting data to the user, or requesting input data from the user. Each of these interactors is bound to exactly one element of the corresponding user interface socket definition. All interactors are sufficiently abstract to ensure that presentation templates can be used in any delivery context, meaning they are modality-independent. They can be ordered as well as semantically or hierarchically grouped within the PreT.

Interactors specified by the PreT mark-up language fall into one of the following categories:

- Input interactors: Represent a request for user input.
- Output interactors: Used to present immutable data to the user.

- Command interactors: Bound to a command defined in the user interface socket; used to activate a command.
- Notification interactors: Suspend the normal interaction to present information, warnings or errors.

The PreT does not include resource information (labels, help texts, keywords etc.) as they are part of the resource description (ISO/IEC 24752-5)

When using a URC-compliant user interface device to control a URC-compliant target service, the UI device would request the presentation template from the device and directly render and display a user interface based on the various definition documents.

Due to lack of adoption, the “Universal Control Hub” [58](UCH) was introduced as an intermediate solution. The UCH architecture is an extension to the URC standard, which introduces a URC-based gateway between target services and user interface devices, allowing end devices that are not URC-compliant to communicate as proposed in the URC specification. This approach therefore circumvents the restriction that direct communication is only possible when all target devices and all UI devices implement the URC specification.

Using UCH, the gateway transforms the presentation template delivered by the service to a concrete user interface, written in any programming or user interface description language that the user interface device’s platform can interpret. The only precondition for this transformation process is that the gateway knows how to map abstract interactors to elements in the target language, or that an external source of information is available that describes this mapping. The concrete user interface description is then delivered to and renderer and displayed by the UI device. Similarly, when the target service delivers data to be presented to the user, or requests data from the user, the data is first processed by the gateway and transformed to formats readable by the respective device.

Level of abstraction

The level of abstraction is quite high, although there are some restricting factors concerning both user interface sockets and presentation templates:

1) Theoretically, presentation templates are modality-independent as the interactors they consist of are sufficiently abstract to be used in any delivery context. In practice, however, the set of proposed interactors seems to be quite closely related to graphical user interfaces. One example concerns the input and text area interactors: To distinguish between short and long string data types may be useful in several delivery contexts, however the above-mentioned interactors use the number of lines to express this distinction (single-line text input is accomplished by using the input interactor, multi-line text input is accomplished by using the text area interactor). When using voice-based interaction, for example, this criterion is not available since spoken text is not divided into lines, therefore in such an environment there is no difference between the input and the text area interactor.
2) There is no guarantee that user interface sockets are platform-independent, due to the potential use of mapping elements in the socket definition.

Adaptability

The main goal of the URC standard is to let the user choose which device to use as remote console for different target devices, which is an important step towards accessibility (especially when compared to traditional systems that force the user to use a specific remote control). However, an important improvement would be the option of

defining preferences on how user interfaces are a specific device (for example, using a high contrast on graphical UI devices, or a certain level of loudness on voice-based interaction systems). The URC specification does neither incorporate a user preference definition system, nor other interfaces for external user preference descriptions.

Adaptability concerning different use-cases or contexts of use is currently not supported by the URC framework.

Openness

The URC Standards Whitepaper [87] provides a basic overview of the elements of the first URC specification that formed the ANSI/INCITS 389-393 family of standards, but detailed information on the current (2008) version is only available from the ISO/IEC 24572 standard documents which are not freely available.

Status

The approach was proposed by the INCITS standardisation organisation in the US, approved by the ISO international standardisation organisation, and is currently promoted and implemented by an international consortium containing research organisations, universities and industrial businesses. The URC specification was approved as a US national standard in 2005 and as international standard in 2008, implementation work is currently in progress by the URC consortium.

A reference implementation of the UCH architecture that is based on the URC framework is available for the Java and C/C++ programming languages, developed by the University of Wisconsin-Madison. Based on this reference implementation, several prototypical tools were developed.

The prototypes developed by the URC consortium aim at use with mobile devices and desktop computers.

3.2.2 Extensible Interface Mark-up Language (XIML)

The Extensible Interface Mark-up Language (XIML) is a framework for defining abstract user interfaces and transforming them to concrete ones. In addition, it offers several methods for the automatic adaptation of user interfaces, based on different criteria. Its main focus is the use case of a generic user interface, defined once for an application that can be executed on a variety of devices, using different platforms. [59]

XIML is an XML-based mark-up language. It specifies five basic components that together build the core of an XIML user interface description:

- User component: Defines all users and groups of users that may operate the system. Information is hierarchically structured. Characteristics are defined as key-value pairs.
- Task component: Contains all parts of the business logic that require user interaction, and leaves out business logic beyond the scope of user interaction. This means it defines all tasks that the user may wish to accomplish through the user interface. Information is hierarchically structured.
- Domain component: Contains a hierarchically structured collection of all data objects the user can view or manipulate. It resembles a simple ontology, stored as key-value pairs.
- Dialog component: A dialog component is the equivalent to the task component on the level of concrete user interfaces. It contains a structured set of interaction items.

- Presentation component: The presentation component contains all concrete user interface elements that are available in one concrete, platform-dependent user interface. Each concrete user interface element references a UI widget present in a particular widget toolkit. Therefore an XIML interface definition contains several presentation components, one for each target end device that shall be supported.

XIML allows defining custom elements as content of presentation components, instead of using toolkit-dependent widgets, to overcome the effort of creating specific presentation components for each target platform. Presentation components that consist of such custom elements require one additional transformation step before rendering, but can be used in several XIML projects. Thus they reduce effort.

The component-driven architecture of every XIML interface definition guarantees a strict separation of business logic, interaction definitions, and the rendering of concrete user interfaces.

The different components present in an XIML interface definition are related to each other in different ways. XIML does not provide a predefined, closed set of relations but allows the definition of custom relations. These relations can then be used to model connections between elements defined inside the same component as well as elements of different components. For example, relations can be used to state that certain presentation widgets (particular elements of presentation components) can be used to display a certain data type (a particular element of a data component).

Level of abstraction

XIML's strict separation of business logic, interaction description, and UI rendering is a crucial factor for a high level of abstraction and therefore support for many different target platforms and modalities. Although the core XIML language is modality-extensible rather than modality-independent (as a new presentation component must be created manually for each additional modality to be supported), this drawback can be overcome by using the concept of intermediate presentation components, as explained above.

Adaptability

XIML provides several methods for automatic adaptation of user interfaces, including both built-in functionality and interfaces for external solutions. The specification of user profiles and users' preferences is even one of the five core components of the language, namely the user component. Personalisation is integrated into the basic XIML language, meaning the automatic exchange of user interface widgets that present the same information in different ways based on user preferences. Finally, the mechanism of automatically generating rules for mapping intermediate presentation elements to concrete UI widgets provides a flexible way to react to contextual settings, device capabilities, use-cases, user preferences, etc.

Openness

An “XIML Starter Kit”, including the full XIML language specification, documentation, samples, and tools is available for free after registration from the developer team's web page [102].

Status

XIML was developed by the RedWhale Software Corporation, a software development company specialised on user interface design concepts.1.0 is the latest version, and it was published in 2002. The current development status is unknown.

The “XIML Starter Kit” contains two converters: one for HTML and one for WML (Wireless Mark-up Language) output. Wireless Mark-up Language was developed as part of the Wireless Application Protocol (WAP). It is used for the description of web content pages to be displayed on old mobile devices that are not capable of rendering HTML or XHTML content [89].

The HTML converter mentioned above was developed for use with desktop PCs, and the WML converted was intended for use on mobile devices such as PDAs.

3.2.3 Extensible Interaction Scenario Language (XISL)

XISL is a language for describing online multi-modal interaction systems, using an XML-based syntax. As the name suggests, it describes interaction scenarios rather than concrete user interfaces, thus being applicable to many different interaction modalities. XISL is based on existing open standards such as VoiceXML [60] and SMIL [61], but advances their concepts in a modality-independent approach.

The basic purpose of an XISL document is to describe any web-based task a user might wish to accomplish, independent of the terminal that is used for accessing the web. This is done by precisely specifying the control flow between user and system, including both the exact data that is exchanged and the direction in which the data is transmitted (either from the system to the user, or vice versa). In addition, the exact order in which these data transfers take place must be specified. All this can be encoded as a sequence of interactions between user and system. Finally, those interactions must be specified in a modality-extensible way in order to support any potential end device. [62]

In contrast to modality-independent approaches that define interactions in a way that can be interpreted and rendered on different modalities, XISL follows a modality-extensible approach: New input or output elements are added for each modality an interaction shall support.

XISL introduces a dialog manager that acts as a middleware between web services and user interface devices. The dialog manager ensures that the system is terminal-independent and any end device can be used as terminal to the system. It contains a XISL interpreter that parses XISL documents and is responsible for executing the correct interactions at the right time.

When executing an interaction, the dialog manager sends the content of an input or output element to the current user interface device. The user interface device need not include an XISL interpreter as there is no XISL code for it to parse. The user interface is rendered there based on the platform-dependent code contained in the input or output element, and user input is transmitted back to the dialog manager which transforms it, if necessary, to be readable by the web service.

Level of abstraction

The concept of describing web service access as a sequence of interaction scenarios is quite generic as it does not include any modality-specific details. The same applies to the application flow model that groups one input and one output interaction together and provides mechanisms for executing these interaction scenarios in any order, depending on user input, application state, etc. When it comes to a detailed definition of input and output operations, the concept of the XISL language turns to a very concrete one, relying on separate definitions for each modality. As a consequence, the level of abstraction of XISL must be defined as medium: It offers rather abstract interaction description that needs to be manually extended to be used in practice.

Adaptability

The concrete implementation of input and output operations is left to custom extensions; these are responsible for rendering the user interface and adapting it to users' needs or environmental settings. The XISL core language does not provide any framework for automatic adaptation. The same applies to use-case awareness: The modality-extendable system allows input and output interactions to be presented in quite different ways using different modalities, but the execution of different interaction scenarios depending on the currently used modality is not provided.

Openness

The full specification of both the core language and a front end extension is available freely on the developer's web page; the latest version of both specifications is available only in Japanese language however.

Status

XISL was developed by a research group of the Toyohashi University of Technology in Japan. Version 1.0 (which was called “Extensible Interaction Sheets Language”) was published in January 2002; the updated version 1.1 was published in April 2003. This is the latest version; the current development status is unknown.

A full specification for PC, mobile phone and PDA platforms exists as extension to the core XISL language, published by the same research group that developed the original XISL language [90]. This extension specifies the content of input and output elements for these platforms. In addition, a runtime environment was developed for XISL as part of the Galatea Project [91] which provides a speech-based input and output system, featuring animated virtual avatars.

The front end extension mentioned above covers traditional PCs and mobile devices, and the Galatea project features speech-based input and output, totalling in three different target platforms.

3.2.4 Web Service Experience Language (WSXL)

The Web Service eXperience Language (WSXL) was developed to reduce development effort while building web applications for different distribution channels by re-use. Service-based web applications specified using WSXL can easily be adapted to different presentation scenarios.

WSXL is based on or closely collaborating with existing, widely-used web standards, such as the SOAP protocol, the Web Services Description Language (WSDL), and the XForms standard [63].

The WSXL specification [64]resembles the widely-used Model-View-Controller pattern, as it separates presentation, data, and control in particular components. A WSXL application consists “*of one or more data and presentation components, together with a controller component which binds them together and specifies their interrelated behaviour”* [65].

- Base component: All types of WSXL components are derived from this component, and therefore share basic operations specified within. I.e. live cycle operations, or export operations.
- Presentation component: represents a complete user interface (e.g. an HTML page) and all of the user interface elements. WSXL does not restrict the set of allowed UI elements, nor provide a set of default UI elements. For each target platform that

shall be supported, one presentation component has to be defined, using UI widgets that are supported by the particular platform.

- Data component: Contains all data stored by the web application, available to the user interface. Data components may be bound to presentation components by using control components, or external data sources.
- Control component: Manages binding between presentation and data components, by invoking event handlers in order to synchronise changed data between Presentation and Data component.

WSXL includes an “Adaption Description Language”, to control several aspects of the automatically generated final user interfaces through so called adaption points. Such an adaption point refers to an element in a data or presentation component via XPath [66] or XQuery [67]. The adaption point specifies one of the three operations, “lookup”, “insert”, or “replace”, to indicate that the operation is permitted on the UI or data element. This means a third party developer may adapt the rendering of the element.

Level of abstraction

Although WSXL is designed for web services and their typical delivery channels (HTML pages and Flash applets in web sites, portals, etc.), the level of abstraction is quite high. By defining separate presentation components for each output toolkit and just binding them to the underlying data components, theoretically every toolkit and therefore every output modality is supported. The approach, however, does not really generate final user interfaces automatically; a concrete user interface must rather be created manually for each toolkit the web service shall support.

Adaptability

WSXL provides a powerful adaptability description framework, allowing in-depth specification for both service providers as well as distributors. Parts of user interfaces may be explicitly authorised for adaptation, and those parts may be adapted directly on code-level while all the others stay closed. This concept was designed for adaptation to different distribution channels inside the usage context of web services. Therefore no built-in support for the automatic adaptation following environmental contexts, use cases, or user preferences is provided, although it could easily be added by external toolkits.

Openness

The full WSXL specification of both the current language version and the former, deprecated version is freely available from the developer's web page [92].

Status

WSXL was developed by IBM.

The first language version was published in October 2001; version 2 was published in April 2002. Version 2 is the latest version; no updates have been published since then by IBM. The current development status is unknown.

Although the exact language specification is freely available, there exists no public information about the use of WSXL in practice. The language specification provides examples of WSXL implementations for two target toolkits: HTML and Adobe (former Macromedia) Flash.

3.2.5 User Interface Extensible Mark-up Language (UsiXML)

The USer Interface eXtensible Mark-up Language (UsiXML) is an XML-based mark-up language that can be used to define user interfaces in an abstract way, independent of modalities. Its main goal is to reduce the effort necessary for generating user interfaces to be used in multiple contexts of use.

UsiXML defines user interactions on four layers resembling transformational steps from abstract to concrete user interfaces, based on the Cameleon reference framework [68]: The layers are [69]:

- Tasks and concepts: Abstract task descriptions describe the tasks a system offers to a user. They define all interactions possible, as well as objects manipulated. On this layer, three types of models are defined:
    - Task models: Contain tasks and task relationships.
    - Domain models: Describe real-world concepts that tasks resemble from the user's point of view.
    - Context models: Describe environmental settings that influence the execution of a task. This includes user models that assign preferences to a group of users, platform models that comprise capabilities of user interface devices, and environment models that describe all external influences (e.g. physical or psychological conditions).
- Abstract user interfaces: define user interactions in a modality-independent way, based on an underlying task description. An abstract UI consists of *abstract interaction objects* (AIO) that represent modality-independent forms of traditional UI widgets. Each AIO is assigned one or more functionalities, including input, output, navigation, and control functionalities.
- Concrete user interfaces: concretise abstract UIs to work with one specific input/output modality. In summary, they define the look and feel of a certain user interface, in a modality-dependent but toolkit-independent way. Concrete UIs are composed of *concrete interaction objects* (CIO) that represent UI widgets. For example, when using graphical user interfaces, widgets such as buttons, text boxes, etc. are used as CIOs.
- Final user interfaces: are concrete UIs that are supposed to be run on a specific platform, meaning they are platform- and toolkit-dependent.

To formalize the transformation steps necessary for conversion between two of the above-mentioned abstraction layers, UsiXML proposes a graph-based transformation specification [70].

This graph-based transformation logic uses a tree-based structure: The abstract task description (the first abstraction layer) is used as root of the tree, the corresponding abstract UI forms the second level, and on the concrete UI and final UI layers different branches represent the various UIs for different modalities or platforms, respectively.

All of the four abstraction layers can be further decomposed into sub layers. This means that transformations between two neighbouring layers include several transformation steps. This way, transformations between any of the four layers can be realized by composing multiple transformation steps and carrying them out on after the other.

Carrying out transformations is impossible without detailed mapping information, encoding which elements of the source layer are to be transformed to which elements of

the target layer. This information is included in special rule sets. For example, for each UI toolkit that should be supported (meaning that a concrete UI can be transformed to a final UI for this toolkit), such a rule set must be provided from an external source.

Level of abstraction

Due to the layered structure of UsiXML, the language provides a high level of abstraction. In theory, all modalities are supported since the first two abstraction layers, tasks and abstract user interfaces, are modality-independent. In practice, there are two restrictions concerning device-independence: First, UsiXML is currently designed to support only two types of modalities, as stated in [93]: “Two modalities lie in the intended scope of USIXML: graphical and auditory.” However, due to its structure the language should be quite easy to extend to support additional modalities. Second, only target toolkits are supported that provide a rule set for transformation of concrete UIs to final UIs in that toolkit.

Adaptability

Full adaptability of user interfaces defined in UsiXML is supported. Context models are used to store information about user preferences, device capabilities, and environmental settings, thus offering a complete framework to support automatic accessibility, adaptability to contexts of use, and use-case awareness.

Openness

The full language specification, including documentation, examples, and tools, is freely available from the UsiXML project web page.

Status

UsiXML was developed by a research group with members from several universities and research organisations. The work on UsiXML began in 2004. The latest version 1.8 was published in 2007; current status of development is unknown.

Several graphical editors for creating UsiXML documents were published, including plugins for generating final user interfaces in various target toolkits. One of those editors is GraphiXML [94], which supports XHTML, Mozilla XUL, and Java as target platforms. In addition, several rendering engines for UsiXML documents are available, supporting Adobe Flash and Open Laszlo, among others.

UsiXML is intended for use with only two modalities: Graphical and voice-based user interfaces.

3.2.6 User Interface Mark-up Language (UIML)

UIML is a meta-language, not specifying concrete UI elements on its own, but providing a framework for the definition of custom vocabularies that can then be used to create generic user interface descriptions.

UIML’s main goal is helping UI developers in creating user interfaces that are sufficiently generic to be used on different platforms, thus significantly reducing the effort in developing multi-platform user interfaces. User interfaces defined using UIML are either automatically transformed to different target languages (compiled), or interpreted and rendered on target devices in real-time.

UIML separates a user interface into six parts. The six parts of a user interface description resembled by those six questions build the basic structure of UIML. It is based on the Meta-Interface Model (MIM) suggested by [71], which is defined in a hierarchical manner: On the first level, MIM separates the user interface from underlying application logic and data sources, and from the presentation on specific devices:

- Logic component: Encapsulates the business logic the underlying application provides. It offers the user interface access to the application logic while hiding implementation details such as method and variable names or the protocols used for communication
- Presentation component: Includes information about the rendering of the final user interface, hiding details such as widgets, attributes and event handling.
- Interface component: Allows the description of the communication between user and application in a device- and application-independent way.
    - Structure component: Defines parts that comprise the UI, allowing to group UI elements together and structure them hierarchically.
    - Content component: Defines the content of each part. This may include content types such as text, image, sound, etc.
    - Style component: Defines the exact presentation of each part, including attributes such as font family, text size, colours, etc.
    - Behaviour component: Defines for each part the events, conditions and actions required when interacting with this part of the UI.

UIML is specified [72]as an XML-based meta-language, as it does not contain a predefined set of XML tags that represent UI elements, neither concrete UI widgets nor abstract interactor elements. Instead, it relies on external vocabularies that define a set of abstract UI elements and their mapping to concrete, platform-dependent UI widgets. A UIML document references such a vocabulary, objects defined in the vocabulary may then be used in the UIML document.

Each vocabulary may have a different level of abstraction, depending on the number of target platforms and modalities it shall support. It must define each target toolkit its supports, and each element defined by the vocabulary should include a mapping to one concrete element per toolkit that is targeted. In some cases, an abstract element may map to different concrete elements, or to a certain combination of several concrete elements. Vocabularies that are defined to support only one target toolkit, or a limited set of target toolkits that are similar and use the same interaction modality, may be built less generic than vocabularies that need to support a high number of modalities.

Level of abstraction

The level of abstraction is completely dependent on the vocabulary used by a UIML definition. When using a vocabulary for only one target language, the user interface elements used in UIML may be quite concrete. When defining a vocabulary that shall support a variety of different target modalities, UI elements will automatically be more generic. Unfortunately, the level of abstraction cannot automatically be retrieved from UIML definitions or vocabularies, since even vocabularies covering only one target language might be defined in a very generic way. In addition, the level of abstraction also depends on the number of interface subcomponents that are bound to specific platforms: Support for a certain platform might be very low-level even if the vocabulary supports that platform if there is no specific structure, content and style elements defined in the UIML document.

Adaptability

UIML provides the possibility to define different structures, style sheets, and contents as part of one user interface description. This option can be used for reacting to user preferences, contextual settings, use cases, etc. The mechanism of choosing one of the definitions, however, is beyond the scope of the UIML specification and is left to compilers and interpreters. Therefore UIML provides no built-in functionality for user interface adaptation.

Openness

The current UIML language specification [95] as well as the previous one can be downloaded for free from the web page of OASIS [96], the standardisation organisation that is responsible for publication and further development of the UIML standard. In addition, the specification states that it may be freely implemented by anyone. Previous versions of the specification can be downloaded for free from the former UIML web platform that is now discontinued [97].

Status

Development on UIML was started in 1997 at the Virginia Polytechnic Institute and State University. To further develop the core language and additional tools, a spin-off corporation called “Harmonia” was founded. The UIML standard has been adopted by the Organization for the Advancement of Structured Information Standards (OASIS). The OASIS UIML committee is now responsible for further development concerning the UIML standard.

The first version of UIML was published in 1997, the language has been further developed since then and refinement is still in progress. The latest version is 4.0which was published as a final specification in 2009.

Many different UIML compilers and interpreters have been developed in the last 13years, covering a variety of target language and toolkits. For some of those, several implementations are available. Supported target languages and toolkits contain HTML, Java, C++, .NTE, QT, Symbian, WML, VoiceXML, and others.

The above mentioned implementations cover at least four different platforms: Desktop PCs, mobile devices, multimedia devices (TVs) etc., and speech-based systems. However, there may be other implementations supporting additional platforms.

3.2.7 Dialog and Interface Specification Language (DISL)

The Dialog and Interface Specification Language (DISL) [73]is an extended subset of the UIML language specification. It provides a modelling language for specifying dialog models in an abstract way that can be used to generate user interfaces for multiple modalities and platforms. DISL follows the approach of separation of control model and dialog model. The control model contains all data that represents the application state, while the dialog model is responsible for presentation and interaction.

The dialog model proposed by DISL contains three parts:

- Dialog flow: The central component. It controls the data flow inside the user interface components. The dialog flow is a sequence of four steps that are repeated periodically and are carried out by the two remaining components:

  1. Generate a dialog

  2. Present a dialog

  3 Capture user interactions

  4. Evaluate the interaction

- Presentation component: Responsible for steps 1 and 2 in the cyclic process presented above. It depends on *generic interfaces* (compare *abstract interfaces* in most other UIDLs) that describe the user interface to be generated in an abstract way and are specified manually prior to the dialog flow process. They consist of generic UI widgets which are modality-independent as they describe only basic operations such as trigger, data input, data output, etc. At runtime, generic interfaces are mapped to distinct modalities by mapping each generic widget to a concrete widget. Finally the presentation component presents the generated modality-specific user interfaces to the user using a target device.
- Interaction component: captures user interactions with the currently presented user interface and processes them. All user input is collected from the different modalities. Based on the widgets that delivered the user input, the generic widgets on which they are based are tracked using reverse mappings. On the generic interface level, properties of generic widgets can be set or events can be triggered, according to the user input.

Step 4 in the cyclic dialog flow is executed by the behaviour controller: The events triggered and properties set by the interaction component are passed on to the behaviour resolver who manages the control model. The control model consists of content elements which represent the current state of the user interface. The behaviour resolver reacts to interactions captured by the interaction component of the dialog model by setting content elements' properties. Immediately after such an edit operation, it asks the dialog model to refresh the UI presentation.

Level of abstraction

Due to the hierarchical structure and the separation of data and presentation, the level of abstraction is quite high. Every target platform and modality is supported if adequate mapping rules are provided.

Adaptability

Adaptability is one of the key issues of DISL. The language is designed to support switching of end devices on the fly; therefore adaptability to different use cases and environmental contexts is a built-in core component. The specification of user preferences is not provided by the language framework, although adaptability to such profiles could easily be accomplished by using the built-in adaptability mechanisms.

Openness

Besides scientific papers, no detailed information about the DISL specification is available.

Status

The DISL language was developed by a research group formed of members of the Paderborn and Kassel Universities in Germany. It was proposed in a research paper published in 2006, the current development status is unknown.

No information about systems running DISL on specific platforms is available.

3.2.8 Model-based Language for Interactive Applications (Maria XML)

MARIA stands for Model-based lAnguage foR Interactive Applications, an XML-based user interaction description language. It mainly focuses on the definition of user interfaces used to access web service functionalities. The language follows a semi-automatic approach for generation of user interfaces: Basic final user interfaces are generated automatically from

abstract user interface descriptions, but developers are given the possibility to refine these concrete interfaces.

MARIA XML introduces two abstraction layers: User interfaces are defined on an abstract level and on a concrete level [34]. In addition, *data models* are defined that represent the underlying data structure, as well as *event models* which represent the underlying application logic.

- Abstract user interfaces: Are independent of modality and focus on interactions between the user and the web service. They consist of abstract interactors which are grouped together and interconnected through relations.
- Concrete user interfaces: Are modality and platform specific, but toolkit independent. Each interactor on the abstract level is mapped to an element of the concrete user interface, corresponding to typical user interface widgets (e.g. button, text box, list box). The mapping of abstract interactors to concrete elements depends on the characteristics of the target platform.

A rule set defines mappings, either between abstract and concrete interactor elements, or between concrete elements and toolkit-specific widgets. There a two ways of specifying transformation rules [74]:

1) Rule sets for general user are defined once and can be reused for several documents. A mapping for each element in the abstract layer must be provided.
2) Developers may refine automatically generated user interfaces, by editing predefined mappings at document instance level. These edited rules are used only once for a certain document, or a subset thereof.

The usual information flow is as follows:

1) An abstract user interface is defined based on the web service's WSDL description.
2) The abstract user interface is transformed to multiple concrete user interfaces, one for each target platform to be supported.
3) Each concrete user interface is transformed to multiple implementations, one for each UI toolkit to be supported.
4) The final toolkit-specific implementations are executed on the target devices.

Level of abstraction

Due to the structural distinction of task descriptions (through WSDL), abstract and concrete user interfaces and final implementations, the level of abstraction is quite high. In theory, every modality and every target toolkit is supported, if adequate rule sets have been defined.

Adaptability

Two potential mechanisms for UI adaptation can be identified: MARIA XML provides the possibility to influence the UI generation by overriding mapping rules in certain cases. This can be used to adapt the generated user interface according to different external factors; MARIA XML however does not offer automatic adaptation mechanisms. Also no language constructs are provided to store user preferences or device profiles.

Alternatively, the mechanism of migratory user interfaces already implements an automatic adaptation based on contexts of use, similar approaches could be used to achieve adaptation based on user preferences and use cases.

Openness

MARIA XML was developed as part of the MARIAE (MARIA Environment) tool which can be downloaded for free after registration from the project web page [98], including source code. The full language specification however is not available online.

Status

The MARIA XML language and the MARIAE tool were developed by HIIS Laboratory, a research group focusing on human-computer interaction of the Italian National Research Council (CNR). No detailed information is available about the MARIA XML language itself; however the first version of the MARIA tool was published in 2010. The latest version1.3.1 was published in August 2011. In contrast to most other UIDLs, it is a relatively young development, although MARIA XML is based on the discontinued TERESA XML language which was developed by the same research group.

The only publicly available use of MARIA XML is the MARIAE (MARIA Environment) tool [98] which helps in developing multimodal user interfaces and applications based on web services. It is unclear how many working rule sets for transforming concrete user interfaces to toolkit-specific implementations exist.

MARIAE includes example scenarios that demonstrate the generation of concrete user interfaces for at least three different platforms: desktop PCs, mobile devices, and voice-based systems.

3.2.9 Voice XML

VoiceXML is an XML-based mark-up language used to specify user interaction with speech-based systems. The language is standardized by the World Wide Web Consortium (W3C). VoiceXML documents allow the specification of speech-based interactions between a system and its user. Those interactions contain data output from the system to the user, and data input requested by the system from the user.

According to [77], VoiceXML interactions may be composed by fragments of the following types:

- Output of synthetic speech prompts
- Output of predefined audio content
- Input and automatic recognition of spoken phrases
- Input and automatic recognition of DTMF key presses

In addition, VoiceXML provides mechanisms for recording speech input, specifying telephony call control detail (e.g., call transfer and hang-up) and controlling the dialog flow. The latter is especially important compared to user interfaces based on other interaction modalities: Using speech-based interfaces, exactly one type of information can be output to the user or requested from the user at a time, in contrast to graphical user interfaces.

A VoiceXML application consists of several VoiceXML documents, each defining one dialog. A dialog consists of several input and output interactions that are carried out. Each document specifies the dialog that shall be carried out next, by referencing the URL of the document containing that dialog.

Dialogs are composed of forms and menus. A menu presents the user with several options to choose from and specifies which dialog to carry on with on each option. A form defines several interactions, each consisting of a prompt that is output to the user, the expected data requested from the user, and rules for evaluating the input data.

Level of abstraction

VoiceXML follows a quite generic approach by specifying interactions between user and system as data items that are sent between the two communication partners. VoiceXML however focuses especially on the exchange of audio content and includes functionality only for processing speech and spoken phrases; therefore the overall level of abstraction is rather low.

Adaptability

VoiceXML does not provide mechanisms to store users' preferences and automatically adapt dialogs based on these preferences, or to react to environmental factors. The issue of different user interfaces on different devices does not apply to VoiceXML, since the language focuses on speech-based interfaces that are presented in the course of one telephone call using exactly one device.

Status

The full specification of all language versions of VoiceXML was published as recommendation by the W3C and is available online.

Organizational background:

VoiceXML was originally developed by the VoiceXML forum2, a Consortium founded by AT&T, IBM, Lucent and Motorola. The World Wide Web Consortium (W3C) standardized VoiceXML and currently is responsible for further development of the language. The latest version 2.1 [31] was published in 2007. The main goal of the major 2.0 version published in 2004 was to consolidate and standardize various adaptations of the 1.0 version made by third party developers. Version 3.0 is currently available as public draft.

VoiceXML is used by many implementations of automatic telephony systems on the World Wide Web. VoiceXML is designed solely to support speech-based user interfaces.

3.2.10 Extensible Application Mark-up Language (XAML)

The eXtensible Application Mark-up Language (XAML) is a declarative mark-up language based on XML developed by Microsoft. Its main use is the specification of user interfaces as part of the Windows Presentation Foundation (WPF). Another area of use for XAML is the definition of user interfaces for web applications that build upon the Microsoft Silverlight framework.

[75] gives an overview of the specification of XAML for its main use case: as user interface description language for the WPF framework. The language follows an XML-based structure with custom elements and attributes. XAML is closely coupled with the language framework that uses XAML as UIDL (in most cases .NET).

In general, all classes that are available in the underlying programming language can be used as elements, this includes not only classes that represent user interface components but even custom classes defined by developers inside the application. Properties of those classes can be referenced as attributes. Each XML element defined in a XAML document declares an instance of the class referenced by the element. This concept is mainly used to declare user interface components and their hierarchy; however it can also be applied to non-visual elements that are references by UI components.

[2] http://www.voicexml.org/

XAML is usually compiled to binary code, integrated into .NET applications and executed on end devices through the .NET framework. This ensures that no compatibility or rendering problems can occur. At the same time however it reduces the amount of supported end devices to those that run the .NET framework. The only exception of this rule is the use of XAML in Silverlight web applications. In this case, XAML code can be sent to the browser and is interpreted at client side by the Silverlight browser plug-in.

Level of abstraction

XAML relies on the user interface components specified by the underlying language framework. Since it is mainly used in combination with either WPF or Silverlight and those framework define rather concrete UI widgets (such as button, textbox etc.) that are modality- and toolkit-dependent, the level of abstraction is quite low. Also the implementation details indicate a low abstraction level, since user interfaces specified using XAML can only be rendered on devices supporting the.NET toolkit.

Adaptability

Nearly all attributes of XAML user interface components can be bound to variables and therefore changed at runtime instead of setting them explicitly before compiling. This concept provides the basis for adaptation of existing user interfaces XAML however does not provide mechanisms to store users' preferences or to react to environmental factors. The issue of different user interfaces on different devices is not crucial to XAML, since in most cases a separate user interface for each type of device needs to be manually created anyway.

Openness

XAML was published by Microsoft as an open standard; the full specification is available online.

Status

XAML is developed by Microsoft. A separate XAML version is released with each new version of the .NET framework as well as with each Silverlight version. The latest versions are .NET 4.5 (published only as developer preview in 2011) and Silverlight 5 (published as release candidate in 2011). It can be assumed that XAML will be further developed with newer versions of .NET.

Support for XAML is built into WPF and Silverlight, as mentioned above. Beside those two platforms, there exists the Mono project that aims at developing an open source, cross platform implementation of the .NET framework. The Olive project [99] provides add-on libraries to the Mono framework, including libraries that add XAML support. The Moonlight project [100] aims at developing an open source implementation of Silverlight as part of Mono.

Both .NET and Mono are available for desktop PC as well as mobile platforms. In addition, XAML supports web-based platforms through Silverlight.

3.2.11 XML User Interface Language (XUL)

The XML User interface Language (XUL) is a user interface description language developed by Mozilla. It is based on XML and not compiled but interpreted at runtime by a special rendering engine.

According to [76], XUL separates the description of user interface objects and of these objects' styles. Similar to HTML, in XUL user interfaces are defined in a hierarchical tree-like structure, as some objects may contain others. Since the user interface description is

not compiled but interpreted, the rendering engine builds a document object model (DOM) and hierarchically renders its elements.

XUL provides a predefined set of user interface widgets. XUL supports only graphical user interfaces, which allows the set of UI widgets and their available attributes to be relatively small and concrete. In addition, HTML elements can even be used as part of XUL documents just as native XUL user interface widgets. This can be useful for referencing elements XUL does not provide, such as tables for creating advanced layouts, and for embedding Java applets and similar external content.

The application logic is defined in JavaScript documents that are referenced in the XUL document. Each XUL user interface widget provides several event listeners that can be assigned methods defined in the referenced JavaScript. Alternatively, such method invoking commands can be put in special command objects, and these commands can then be referenced inside an event handler instead of calling JavaScript methods directly.

Level of abstraction

XUL focuses on graphical user interfaces and provides widgets that can only be used in graphical environments; therefore the level of abstraction must be defined as low.

Adaptability

XUL does not integrate mechanisms for defining user preferences or contexts of use, therefore XUL-based user interfaces cannot automatically adapt to those settings, although adaptation could be integrated using the flexible CSS styling system that allows redesign of all user interface elements at runtime.

XUL user interfaces automatically adapt to different hardware capabilities such as screen resolutions. However this applies only to simple dynamic resizing and positioning of widgets, since XUL is restricted only to graphical user interfaces. In addition, XUL does not support the automatic display of different user interfaces (accomplishing different tasks) on different devices, since there is no way to define the type of tasks each widget implements.

Openness

The full specification is available at the Mozilla project web page.

Status

XUL is developed by the Mozilla open source community. XUL is used as user interface description languages by all applications published by Mozilla, the most common ones being the Firefox web browser and the Thunderbird E-Mail client. There are some third-party applications however which also make use of the XUL language; among them the movie/theatre project management tool CeltX and the instant messaging client Instantbird. In addition, the XUL language is used to define user interfaces of Firefox and Thunderbird add-ons. Finally, XUL can also be used to develop applets to be embedded into web pages [101]. This approach however is not very common among web developers, mainly due to the restricted browser support (XUL applets can only be run in Mozilla-based

Currently there is only one XUL interpreter available, namely the Gecko rendering engine developed by Mozilla. This rendering engine is also used in Mozilla-based browsers to render HTML content. Standalone XUL application are interpreted and rendered by the XULRunner engine, which internally uses the Gecko rendering engine.

XUL was developed solely to support graphical user interfaces.

3.3 Results of the UIDL Analysis

All criteria for the suitability of the UIDLs in the AALuis context were assessed. Table 3 gives an overview of the assessment results. Single criteria were rated as inadequate (--), somewhat inadequate (-), indifferent (0), somewhat adequate (+), adequate (++), or unknown (?). Each + was given a rating of 1, each - was given a rating of -1. All UIDLs with a positive total rating were ranked and all ranked UIDLs shall be considered for application in the further specification process of the AALuis architecture.

Criterion	*URC*	*XIML*	*XISL*	*WSXL*	*UsiXML*	*UIML*	*DISL*	*MariaXML*	*VoiceXML*	*XAML*	*XUL*
Abstraction	-	+	0	+	-	0	++	++	--	--	--
Adaptability	--	++	--	+	++	-	++	-	--	0	-
Openness	--	++	+	++	++	++	?	++	++	++	++
Status	++	--	0	-	+	++	?	++	+	++	++
Total rating	-3	3	-1	3	4	3	4	5	-1	2	1
Ranking	N/A	**4**	N/A	**4**	**2**	**4**	**2**	**1**	N/A	**7**	N/A

Table 3: UIDL Assessment
Assessment of the suitability of the analyzed UIDLs in the AALuis context.

4 Enabling Freedom of Choice of User Interfaces

One of the goals of the AALuis project is that every person can use his preferred kind of user interface for any connected service. One precondition to enable freedom of choice of user interfaces at the user's site is to provide user interfaces as consistent as possible across devices taking into account user needs and preferences as well as modalities of the devices. In this section we present attributes and techniques of consistent user interfaces and the necessary adaptation processes in the background.

4.1 Consistency of Multiple User Interfaces

4.1.1 Three-dimensional model of interface consistency

Consistency is not only an issue when designing for various devices, the principle of consistency also applies for user interfaces for one device and one application. Every set of usability guidelines and heuristics strives for consistency (see section 2.1.3 or e.g. the influential publications of Nielsen 1994 [29] and Shneiderman 1997 [45]). For a better understanding of the concept of consistency for websites Ozok and Salvendy (2000) [32] classified it into three types and described the so called three-dimensional model of interface consistency:

- **Conceptual consistency** (language, stereotypes, task concept, skill transfer, output consistency, hierarchical order of concept, etc.): Leaving something to users' interpretation due to lack of explicitness leads to a wrong mental model of the user.
- **Communicational consistency** (moving between screens, menus, user conventions, between-task consistency, distinction of tasks and objects, etc.): It deals with how the user interacts with the user interfaces and whether the means of interaction are consistent for fulfilling the same or similar tasks.
- **Physical consistency** (colour, size, shape, location, spacing, symbols, etc.): It summarizes the consistency of the visual appearance of the user interface and indicates that the features are supposed to be consistent with the users' mental model.

4.1.2 Multiple User Interfaces and Consistency

Multiple user interfaces (MUI) provide different views of the same information and coordinate the services available to users from different computing platforms (Seffah and Javahery 2004 [44]). Pyla et al. (2006) [37] questioned on which level (UI, Task or Data) consistency for MUIs should apply and argued that consistency in UI design should be followed only when it helps reduce the cost of task disconnects (the gap during the change of the devices) and helps support seamless task migration. In a follow-up study they found several problems of multi-device interaction (Pyla et al. 2009 [38]):

- Dropping the use of multiple devices in favour of a single one
- High costs of remembering file and data locations
- Fear of making errors due to overheads associated with the migration of information across devices
- Keeping track of version information

To overcome these problems they built a prototype for engineering software for various devices taking into account these issues and evaluated it successfully. So it is more important to achieve consistency at the mental model and data model level and not just at

the interaction level. MUIs can thus support different types of look-and-feel and offer different interaction styles. However, these have to take into account the constraints of each computing platform while maintaining the following aspects:

Cross-platform consistency [44]: All user preferences must be preserved across the various platforms.

Example: If the end-user has specified a particular access mechanism using one user interface, it should be used on all user interfaces.

Abstraction and Migration [37][44]: All data and services should be the same across platforms supporting the same level of interactivity (even if not all data and services are shown for all platforms) before interaction can proceed seamlessly on various devices.

Example: a product listing might include only the best-selling items on a handheld device, with the rest relegated to a secondary "more products" page. For an office desktop, the product list includes all the items for sale.

Uniformity [44]: A MUI should offer support for the same functionality and feedback even if certain features or variations are eliminated on some platforms.

Example: An airline reservation system presents choosing a flight and buying the ticket in two separate steps. This separation should be preserved on all versions instead of unifying the two tasks into a single step on a simplified version of the interface.

Intra-platform consistency [37][44]: It is not necessary for all features to be made available on all platforms. The application developed for each platform must stay consistent with design guidelines for that particular platform.

Example: A PDA interface could eliminate images or it might show them in black and white. Similarly, text can be abbreviated on a small display, although it should be possible to retrieve the full text through a standardized command.

Holistic Interaction Design [37]: All platforms need to be considered together and functionality needs to be distributed or replicated according to the affordances and contexts of use of each device.

User awareness of trade-off [44]: It would be acceptable to have a simplified version of a program that excludes certain less-important features *(Example: such as specifying a seating preference)* that are present in the more advanced version. Missing these features is a trade-off that the user would be willing to make in return for the benefits of being able to use the system in mobile contexts.

4.1.3 Usability and UX Assessment of MUIs

4.1.3.1 Taxonomy of usability factors for MUIs (Öquist et al. 2004[31])

It is impossible to account for all factors that affect usability in mobile contexts, but by monitoring a few of them it should be possible to make some well-founded predictions about usability. Öquist et al. singled out four factors that have a large impact on mobile usability:

- The first one is **portability**, which embraces in which contexts of use a certain device or interface may be usable. The values plural (usage of several devices), dual (usage of two devices) and single (focus on one device) are possible.

- The second factor is **attentiveness**, which is based on how much attention a certain interface can take for granted under various circumstances. The values primary (main focus onto device), secondary (main focus onto primary task but interaction possible by other modality) and minimal (main focus onto surroundings) are possible.
- The third factor is **manageability**, which is determined by how different interfaces can be handled by the user. Different levels of manageability are defined according to the stability that is achieved: thus there is two-handed stable (desktop environment), two-handed unstable (mobile device held in one hand and operated with the other hand) or one-handed unbalanced manageability.
- The last factor is **learnability**, which has to do with how easily an interface can be learned. Three design paradigms can be distinguished: technological, metaphorical, and idiomatic. The technology paradigm builds on understanding, whereas the metaphoric paradigm is based on intuition, and the idiomatic paradigm is based on providing a swift learning environment to accomplish tasks.

Context of use	*Factors of usability*			
	Portability	*Attentiveness*	*Manageability*	*Learnability*
Stationary	Plural	Primary	Stable	Technological
Seated	Plural	Primary	Stable	Metaphoric
Standing	Dual	Secondary	Unstable	Metaphoric
Moving	Single	Minimal	Unbalanced	Idiomatic

Table 4: MUI Usability
Taxonomy of usability factors for MUIs (Öquist et al. 2004 [31])

This usability assessment method can be extended to assess usability over other MUIs as well. The most straightforward way of doing so is to add additional contexts of use, distinguish the usability factors that are pertinent to them, and create new indexical templates for each contexts of use to match against [31].

4.1.3.2 Multi-Device UI analysis grid (Denis and Karsenty 2004 [6])

It is important for users to be able to easily transfer and adapt their knowledge of a service and task representation from one device onto another. Due to different design constraints between devices certain obstacles arise. Consistency of design should be promoted when possible while certain inconsistencies due to operational constraints, utility and efficiency criteria need to be allowed. The functionality should even help users to understand the limitations imposed by the devices [6]. Table 5 depicts the accordant design principles.

Design Principle	*Dimensions of inter-usability*	
	Knowledge Continuity	*Task Continuity*
Inter-device consistency		
Transparency		
Dialogue adaptability		

Table 5: Multi device analysis grid
Multi-device systems analysis grid (Denis and Karsenty 2004 [6])

Inter-device consistency – Factors:

Perceptual consistency: When possible, the appearance and structure of the information should be similar on different devices. With graphical interfaces, this similarity applies to objects and the spatial organisation of information. With a voice interface, similarity applies to the order in which the information is presented – this order should be consistent with the order in the visual interface(s).

Lexical consistency: The objects of the user interface should have the same label across devices.

Syntactical consistency: To attain a given goal, the same operations should be performed across devices. It is possible to modify a task slightly for a different device in order to make it more efficiently adapted to that device.

Semantic consistency: Services should be similar across devices. In other words, the partition of data and function should be redundant between devices. In the same way, the effect of the operations should be as similar as possible across devices. To ensure task continuity, the state of the data in the last operations performed by the user should be reflected on all devices. Moreover, so as to recover the context of their tasks, users should be able to recover the state of their activity and/or the history of their last operations.

Transparency:

Transparency can be defined as a property of the man-machine dialogue allowing users to construct an accurate representation of the system so as to interact efficiently with it. This property can be based on guidance methods appropriate to the user's expertise level and system functionalities. Adapting transparency to familiar devices and procedures requires the construction of a centralised user model, able to be updated and consulted from each device.

Adaptability:

System transparency is a dynamic notion since it depends on the user's representation of the system, which itself evolves in time and varies with the context of use. Adaptability is the process to adapt a multi-device user interface to the user's model. For *knowledge continuity*, adaptation mainly involves varying the guidance level and amount of explanation of how the system works and the service limits for a given device. For task continuity, system adaptation can involve the contextualisation of data, particularly by reminding the user of the actions that originated the current state of the data.

4.1.3.3 Initial Framework for cross-platform UX (Wäljas et al. 2010 [51])

Based on the work of Denis and Karsenty (2004) [6] and other prior work Wäljas et al. (2010) [51] constructed an initial framework for cross-platform service UX. The framework conceptualizes a structured set of distinct, designable characteristics of cross-platform systems that essentially influence UX, and the respective main elements of cross-platform service user experience (see Figure 5).

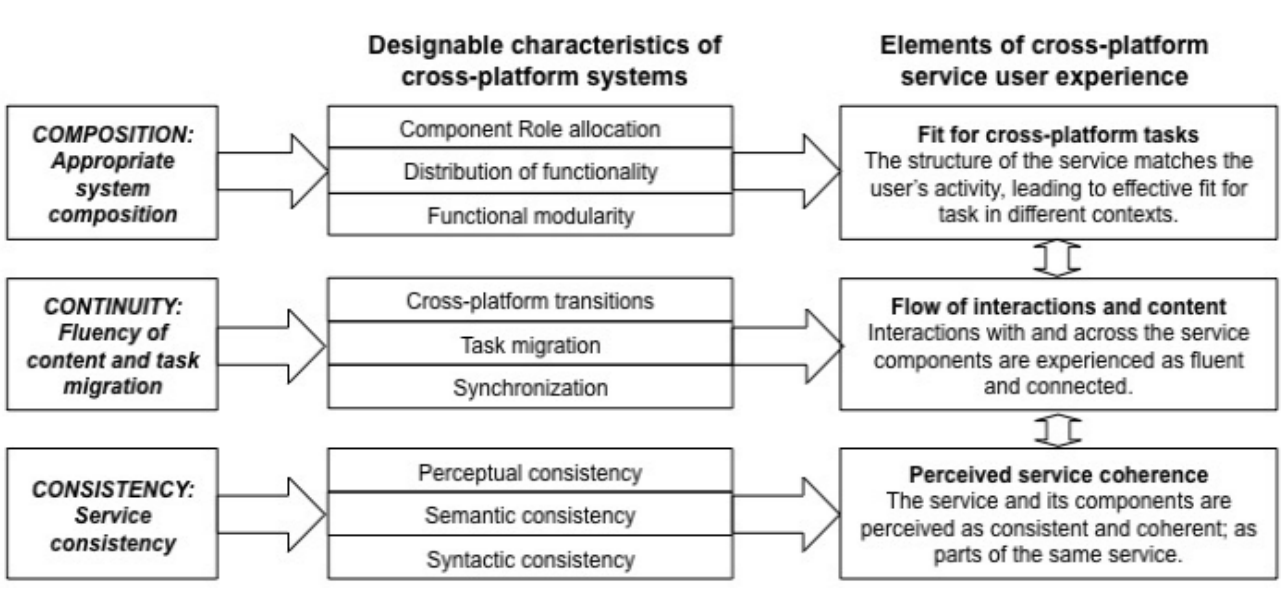

Figure 5: Cross platform service UX
Initial Framework for cross-platform service UX (Wäljas et al. 2010 [51])

Composition:

Component role allocation defines how users perceive the purpose of each system component. Users allocate roles through their use practices: task-based and situation-based role allocation can be distinguished. Task-based allocation means the use of distinct platforms for distinct tasks, whereas situation-based allocation means using distinct devices for the same tasks, but in different situations.

Distribution of functionality means that not every task or content respectively needs to be included in every device. For example, if it is known that a specific device is only used in certain kinds of situations, it may be justified to limit its functionality to support only those situations.

Functional modularity should be maintained to some degree, even though devices in a system are specialised. The degree of functional modularity determines how each platform adapts to use in different situations.

Continuity:

Cross-platform transitions include interactions where the user switches from using one device to using another.

Appropriate task migration: In multichanneling, supporting repetition of tasks is in focus. The same content and functionality needs to be available on all platforms that are used for carrying out the task. With cross media systems, a logical chain of tasks needs to be supported.

Synchronization of actions and content is especially important for mutlichanneling: Users expect to see the exact same content and state of actions when migrating their tasks from one platform to another.

Consistency:

The challenges regarding consistency lie in the heterogeneity and constraints of different technologies. Consistency can be leveraged on different levels to promote a coherent system image (see above). *Coherent user experience* is the ultimate goal of consistent

cross-platform service design. System coherence summarizes the experience of interacting with a service through multiple devices.

4.1.4 Design Frameworks for consistent MUIs

Uniform (Using Novel Interfaces for Operating Remotes that Match) is a system that automatically generates consistent interfaces from potentially inconsistent interface specifications (Nichols et al. 2006[28]). Finding particular functions can be a challenge, because appliances often organize their features differently. This paper presents a system, called Uniform, which approaches this problem by automatically generating remote control interfaces that take into account previous interfaces that the user has seen during the generation process. The similarity information allows the interface generator to use the same type of controls for similar functions, place similar functions so that they can be found with the same navigation steps, and create interfaces that have a similar visual appearance

Jelly is a UI design environment for designing cross-device UIs manually: to create UIs for different computing platforms and toolkits inside one design environment which has the ability to share and edit parts of UIs across devices (Meskens et al. 2010[24]).

GUIDE2ux is a UI design environment implemented in Jelly that (i) identifies and shows usability problems automatically and (ii) facilitates designers to verify their designs on the target device. GUIDE2ux wants to make design standards more accessible for designers and help them to improve and test the user experience of their designs (Meskens et al. 2011 [25]).

References

[1] Apted, T., J. Kay and A. Quigley (2006). Tabletop sharing of digital photographs for the elderly. Paper presented at the meeting of the CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA.

[2] Becker, S. A. (2004). A study of web usability for older adults seeking online health resources. ACM Trans. Comput.-Hum. Interact. 11 (4): 387--406.

[3] Bhachu, A. S., N. Hine and J. Arnott (2008). Technology devices for older adults to aid self management of chronic health conditions. In Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility, New York, NY, USA.

[4] Chaparro, A., M. Bohan, J. Fernandez, S. D. Choi and B. Kattel (1999). The impact of age on computer input device use: Psychophysical and physiological measures. Int. 1. Industrial. Ergonomics 24,503-513.

[5] Chisnell, et al. (2006). New Heuristics for Understanding Older Adults as Web Users. Technical Communication, 53, Society for Technical Communication, 39-59.

[6] Denis, C. and Karsenty, L. (2004). Inter-Usability of Multi-Device Systems – A Conceptual Framework, in Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces (Eds. A. Seffah and H. Javahery), John Wiley & Sons, Ltd, Chichester, UK, 373-385

[7] Eurostat (2008). Ageing characterises the demographic perspectives of the European societies, http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-SF-08-072/EN/KS-SF-08-072-EN.PDF
(26-01-2011)

[8] Greenstein, J.S. and L.Y. Arnaut (1988). Input Devices, in Handbook of Human-Computer Interaction, M. Helander, Editor. 1988, North Holland: Amsterdam

[9] Greil, H., Voigt, A., Scheffler, C. (2008). Optimierung der ergonomischen Eigenschaften von Produkten für ältere Arbeitnehmerinnen und Arbeitnehmer–Anthropometrie. Bundesanstalt für Arbeitsschutz und Arbeitsmedizin

[10] Guerreiro, T., P. Lagoá, H. Nicolau, P. Santana and J. Jorge (2008). Mobile text-entry models for people with disabilities. Paper presented at the meeting of the ECCE '08: Proceedings of the 15th European conference on Cognitive ergonomics, New York, NY, USA.

[11] Häikiö, J., Wallin, A., Isomursu, M., Ailisto, H., Matinmikko, T., & Huomo, T. (2007). Touch-based user interface for elderly users. *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services* (pp. 289–296).

[12] Hawthorn, D. (2000). Possible implications of ageing for interface designers. Interacting withComputers, 12:507–528.

[13] Holzinger, A., Stickel, C., Fassold, M., & Ebner, M. (2009). Seeing the System through the End Users ' Eyes : Shadow Expert Technique for Evaluating the Consistency of a Learning Management System. *HCI and Usability for e-Inclusion* (pp. 178–192). Springer.

[14] Hollinworth N. (2009). Improving Computer Interaction for Older Adults. SIGACCESS Newsletter (93), Jan 2009:1117

[15] Iglesias, R., Gomez de Segura, N., and Iturburu, M. (2009). The elderly interacting with a digital agenda through an RFID pen and a touch screen. In Proceedings of the 1st ACM SIGMM international Workshop on Media Studies and Implementations that Help Improving Access To Disabled Users

[16] Jacko, J., Emery, V.K., Edwards, P.J., Ashok, M., Barnard, L., Kongnakorn, T., Moloney, K.P., Sainfort, F. (2004). Effects of multimodal feedback on older adults' task performance given varying levels of computer experience. Behaviour & Information Technology, 23(4):247–264.

[17] Jin, Z. X., T. Plocher and L. Kiff (2007). Touch Screen User Interfaces for Older Adults: Button Size and Spacing, Universal Access in Human Computer Interaction. Coping with Diversity, Springer, Berlin/Heidelberg, pp 933-941.

[18] Kurniawan, S. and P. Zaphiris (2005). Research-derived web design guidelines for older people. Paper presented at the meeting of the Assets '05: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility, New York, NY, USA.

[19] Lee, C.-F. and C.-C. Kuo (2007). Difficulties on Small-Touch-Screens for Various Ages. Universal Acess in Human Computer Interaction. Coping with Diversity: 968--974.

[20] Lepicard, G. and N. Vigouroux (2010a). Influence of age and interaction complexity on touch screen. 12th IEEE International Conference on e-Health Networking Applications and Services (Healthcom) (Lyon, July 1-3, 2010).

[21] Lepicard, G. and N. Vigouroux (2010b). Touch Screen User Interfaces for Older subjects Effect of the targets number and the two hands use. International Conference on Computers Helping People with Special Needs (Vienna, July 14-16, 2010).

[22] Lorenz, A., D. Mielke, R. Oppermann and L. Zahl (2007). Personalized mobile health monitoring for elderly. Paper presented at the meeting of the MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services, New York, NY, USA.

[23] Maguire, M. C. (1999). A review of user-interface design guidelines for public information kiosk systems. *Int. J. Hum.-Comput. Stud.* 50, 3 (March 1999), 263-286.

[24] Meskens, J., Luyten, K., & Coninx, K. (2010). Jelly: A multi-device design environment for managing consistency across devices. Proceedings of the 2010 International Conference on Advanced Visual Interfaces (pp. 289-296)

[25] Meskens, J., Loskyll, M., Seißler, M., Luyten, K., Coninx, K., & Meixner, G. (2011). GUIDE2ux: a GUI design environment for enhancing the user experience. Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems (pp. 137-142)

[26] Moffatt, K. A., McGrenere, J. (2007), Slipping and drifting: using older users to uncover pen-based target acquisition difficulties, Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility

[27] Moffatt, A. K., Yuen, S., McGrenere, J. (2008), Hover or tap?: supporting pen-based menu navigation for older adults, Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility

[28] Nichols, J., Myers, B. A., & Rothrock, B. (2006). UNIFORM: automatically generating consistent remote control user interfaces. *Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 611–620). ACM.

[29] Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY

[30] Norman, D. A. 2010. The way I see it: Natural user interfaces are not natural. interactions 17, 3 (May. 2010), 6-10.

[31] Öquist, G., Goldstein, M. and Chincholle, D. (2004). Assessing Usability across Multiple User Interfaces, in Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces (Eds. A. Seffah and H. Javahery), John Wiley & Sons, Ltd, Chichester, UK, 325-349.

[32] Ozok, A. A., Salvendy, G. (2000). Measuring consistency of web page design and its effects on performance and satisfaction. Ergonomics 43(4), 443–460.

[33] Parhi, P., Karlson, A. K., Bederson, B. B. (2006)Target size study for one-handed thumb use on small touchscreen devices, Proceedings of the 8th conference on Human-computer interaction with mobile devices and services

[34] Park, Y. S., Han, S. H., Park, J., Cho, Y. (2008). Touch key design for target selection on a mobile phone, Proceedings of the 10th international conference on Human computer interaction with mobile devices and services

[35] Perry, K. P., Hourcade, J. P. (2008), Evaluating one handed thumb tapping on mobile touchscreen devices, Proceedings of graphics interface

[36] Potter, R. L., Weldon, L. J., Shneiderman, B. (1988), Improving the accuracy of touch screens: an experimental evaluation of three strategies, Proceedings of the SIGCHI conference on Human factors in computing systems, 27-32.

[37] Pyla, P. S., Tungare, M., & Pérez-Quinones, M. (2006). Multiple user interfaces: Why consistency is not everything, and seamless task migration is key. Proceedings of the CHI 2006 workshop on the many faces of consistency in cross-platform design (pp. 1-4)

[38] Pyla, P. S., Tungare, M., Holman, J., & Pérez-Quiñones, M. (2009). Continuous user interfaces for seamless task migration. Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction (pp. 77–85). Springer.

[39] Rajala, T., Lahtinen, Y., and Paunio, P. Suurten kaupunkien 2. RAVA-utkimus. Vanhuksien toimintakyky ja avun tarve.Suomen kuntaliitto, 2001. (in Finnish)

[40] Roudaut, A. (2009). Visualization and interaction techniques for mobile devices. In Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Extended Abstracts Volume, Boston, MA, USA, April 4-9, 2009, ACM (2009) 3153–3156

[41] Saffer, D. (2008). Designing Gestural Interfaces, OReilly, Cambridge.

[42] Schedlbauer, M. (2007). Effects of Key Size and Spacing on the Completion Time and Accuracy of Input Tasks on Soft Keypads Using Trackball and Touch Input. Human Factors and Ergonomics Society Annual Meeting Proceedings 51: 429-433(5).

[43] Sears, A. (1991). Improving touchscreen keyboards: design issues and a comparison with other devices. Interacting with Computers 3 (3): 253 - 269.

[44] Seffah, A. and Javahery, H. (2004) Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces, in Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces (Eds. A. Seffah and H. Javahery), John Wiley & Sons, Ltd, Chichester, UK, 11-26

[45] Shneiderman, B. (1997). Designing the User Interface. Strategies for effective Human-Computer Interaction, 3rd edn. Addison-Wesley, Reading

[46] Stößel, C. (2009). Familiarity as a factor in designing finger gestures for elderly users. In Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09). ACM.

[47] Sun, X., T. Plocher and W. Qu (2007). An Empirical Study on the Smallest Comfortable Button/Icon Size on Touch Screen, Usability and Internationalization, Lecture Notes in Computer Science: HCI and Culture, Springer Berlin, Vol 4559.

[48] Terrenghi, L., D. Kirk, A. Sellen and S. Izadi (2007). Affordances for manipulation of physical versus digital media on interactive surfaces. Paper presented at the meeting of the CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA.

[49] Tsai, W. C. and C. F. Lee (2009). A study on the icon feedback types of small touch screen for the elderly, Universal Access in HCI, Part II, HCI 2009, LNCS5615, Paper presented at the 13th International Conference on Human-Computer Interaction, San Diego, USA, pp. 422-431.

[50] Vastenburg, M., Visser, T., Vermaas, M., & Keyson, D. (2008). Designing acceptable assisted living services for elderly users. *Ambient Intelligence*, 1–12

[51] Wäljas, M., Segerståhl, K., Väänänen-Vainio-Mattila, K., & Oinas-Kukkonen, H. (2010). Cross-platform service user experience: A field study and an initial framework. Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (pp. 219–228). ACM.

[52] Wobbrock, J.O., M.R.Morris and A.D. Wilson (2009). User-defined gestures for surface computing. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '09). Boston, Massachusetts (April 4-9, 2009). New York: ACM Press.

[53] Wood, E., T. Willoughby, A. Rushing, L. Bechtel and J. Gilbert (2005). Use of Computer Input Devices by Older Adults. Journal of Applied Gerontology 24(5): 419-438.

[54] Yang, T. (2008). Appropriate User Interface for the Elderly. Seminarreport

[55] Yuan, Y., Y. Liu and K. Barner (2005). Tactile Gesture Recognition for People with Disabilities. ICASSP. Vol 5, pp 461 – 464.

[56] Information technology - user interfaces - universal remote console. International Organization for Standardization, ISO/IEC 24752, 2008.

[57] Information technology - user interfaces - universal remote console - user interface socket description. International Organization for Standardization, ISO/IEC 24752-2, 2008.

[58] Gottfried Zimmermann and Gregg Vanderheiden. The universal control hub: an open platform for remote user interfaces in the digital home. In *Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques*, HCI'07, pages 1040–1049. Springer-Verlag, 2007.

[59] Angel Puerta and Jacob Eisenstein. XIML: A universal language for user interfaces. Technical report, RedWhale Software, 2001.

[60] Matt Oshry, Michael Bodell, Paolo Baggia, Daniel C. Burnett, Alex Lee, David Burke, Jerry Carter, Emily Candell, R. J. Auburn, Brad Porter, Ken Rehor, and Scott McGlashan. Voice extensible markup language (VoiceXML) 2.1. W3C recommendation, W3C, 2007.

[61] Dick Bulterman. Synchronized multimedia integration language (SMIL 3.0). W3C recommendation, W3C, 2008.

[62] Kouichi Katsurada, Yusaku Nakamura, Hirobumi Yamada, and Tsuneo Nitta. XISL: a language for describing multimodal interaction scenarios. In *Proceedings of the 5th international conference on Multimodal interfaces*, ICMI'03, pages 281–284. ACM, 2003

[63] John M. Boyer. XForms 1.0 (third edition). First edition of a recommendation, W3C, 2007

[64] Ali Arsanjani, David Chamberlain, Dan Gisolfi, Ravi Konuru, Julie Macnaught, Stephane Maes, Roland Merrick, David Mundel, T. V. Raman, Shankar Ramaswamy, Thomas Schaeck, Rich Thompson, Angel Diaz, John Lucassen, and Charles Wiecha. (WSXL) web service experience language version 2. Technical report, IBM, 2002.

[65] David Chamberlain, Angel Díaz, Dan Gisolfi, Ravi B. Konuru, John M. Lucassen, Julie MacNaught, Stéphane H. Maes, Roland Merrick, David Mundel, T. V. Raman, Shankar Ramaswamy, Thomas Schaeck, Richard Thompson, and Charles Wiecha. WSXL: A web services language for integrating end-user experience. In *Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces*, CADUI'02, pages 35–50, 2002.

[66] James Clark and Steven DeRose. XML path language (XPath) version 1.0. W3C recommendation, W3C, 1999

[67] James Clark and Steven DeRose. XML path language (XPath) version 1.0. W3C recommendation, W3C, 1999

[68] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15:289–308, 2003

[69] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, Murielle Florins, and Daniela Trevisan. USIXML: A user interface description language for context-sensitive user interfaces. In *Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages"*, AVI'04, pages 55–62, 2004

[70] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Vнctor López-Jaquero. USIXML: a language supporting multi-path development of user interfaces. In *Proceedings of the 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with the 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems*, volume 3425 of *EHCI-DSVIS'2004*, pages 200–220. Springer Verlag, 2004.

[71] Constantinos Phanouriou. *UIML: A Device-Independent User Interface Markup Language*. PhD thesis, Virginia Polytechnic Institute and State University, 2000.

[72] James Helms, Robbie Schaefer, Kris Luyten, Jean Vanderdonckt, Jo Vermeulen, and Marc Abrams. User interface markup language (UIML) version 4.0. Technical report, Organization for the Advancement of Structured Information Standards (OASIS), 2009.

[73] Robbie Schaefer, Steffen Bleul, and Wolfgang Mueller. Dialog modeling for multiple devices and multiple interaction modalities. In *Proceedings of the 5th international conference on Task models and diagrams for users interface design*, TAMODIA'06, pages 39–53. Springer Verlag, 2007

[74] Fabio Paternò, Carmen Santoro, and Lucio Davide Spano. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.*, 16:19:1–19:30, 2009.

[75] Microsoft. XAML in WPF - .NET Framework 4. http://msdn.microsoft.com/en-us/library/ms747122.aspx.

[76] Neil Deakin. XUL Tutorial. Mozilla Developer Network (MDN). https://developer.mozilla.org/en/XUL_Tutorial

[77] Dave Raggett. Getting started with VoiceXML 2.0. W3C, 2001. http://www.w3.org/Voice/Guide/.

[78] Holzinger, A., Schaupp, K., Eder-Halbedl, W.: An investigation on acceptance of ubiquitous devices for the elderly in a geriatric hospital environment: Using the example of person tracking. In Computers Helping People with Special Needs, 11th International Conference, ICCHP 2008, Linz,

[79] Dave Bryant. The uncanny valley - Why are monster-movie zombies sohorrifying and talking animals so fascinating? Available at: http://www.arclight.net/~pdb/nonfiction/uncanny-valley.html

[80] Morandell, M.: Day Structuring Assistance for People with Alzheimer's Disease. Master Thesis at the University of Linz, Austria, (2007)

[81] Morandell, M., Hochgatterer, A., Wöckl, B., Dittenberger, S., & Fagel, S. (2009). *Avatars@home - interfaceing the smart home for elderly people*. Paper presented at Lecture notes in Computer Science.

[82] Spierling, U.: Der Avatar: "Ein Wesen, eine Spielgur, ein Medium, oder ein UI-Element?". In: Umhegt oder abhängig? Springer Berlin Heidelberg (2006)

[83] Masahiro Mori (translated by Karl F. MacDorman and Takashi Minato). The uncanny valley. Available at: http://www.androidscience.com/theuncannyvalley/proceedings2005/uncannyvalley.html CogSci-2005 Workshop Towards Social Machansims.

[84] Wu, P., Miller, C.: Results from a Field Study: The Need for an Emotional Relationship between the Elderly and their Assistive Technologies. In: 1st International Conference on Augmented Cognition, Las Vegas (2005)

[85] Wada, K, Shibata, T; Kawaguchi, Y: Long-term Robot Therapy in a Health Service Facility for the Aged - A Case Study for 5 Years: 2009 IEEE 11TH INTERNATIONAL CONFERENCE ON REHABILITATION ROBOTICS, VOLS 1 AND 2 Book Series: International Conference on Rehabilitation Robotics ICORR Pages: 1084-1087

[86] Ian Andrew James, Lorna Mackenzie, Elizabeta Mukaetova-Ladinska: Doll use in care homes for people with dementia; International Journal of Geriatric Psychiatry Volume 21, Issue 11, pages 1093–1098, November 2006

[87] Vanderheiden. G., Zimmermann, G., and Trewin S.Interface sockets, remote consoles, and natural language agents - a V2 URC standards whitepaper.Technical report, URC Consortium, 2005

[88] Tools and Prototype Implementations for the URC Framework http://myurc.org/tools/

[89] Wireless application protocol – wireless markup language specification. Technical report, Wireless Application Protocol Forum, 2000

[90] Front-end Specification of XISL, http://www.vox.tutkie.tut.ac.jp/XISL/XISL_Web_Site_E/XislFESpecE.html

[91] Prendinger, H.Ishizuka, M., Life-Like Characters – Tools, Affective Functions, and Applications. Springer Verlag 2004

[92] Web Service Experience Language, IBM, http://www.ibm.com/developerworks/library/specification/ws-wsxl/

[93] Limbourg, Q., Vanderdonekt, J., Michotte, B., Bouillon, L., Florins, M., Trevisan, D., USIXML: A user interface description language for context-sensitive user interfaces, Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages", AVI'04, p 55-62, 2004

[94] Michotte, B., Vanderdonekt, J., GrafiXML, a multi-target user interface builder based on UsiXML. Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems, ICAS'08 p 15-22, IEEE Computer Society, 2008

[95] James Helms, Robbie Schaefer, Kris Luyten, Jean Vanderdonckt, Jo Vermeulen, and Marc Abrams. User interface markup language (UIML) version 4.0. Technical report, Organization for the Advancement of Structured Information Standards (OASIS), 2009

[96] The OASIS Technical Committee, http://www.oasis-open.org/committees

[97] The UIML Specification, http://uiml.org/specs/index.htm

[98] HIIS Laboratory, The MARIA Environment, http://giove.isti.cnr.it/tools/MARIAE/

[99] Mono: Olive Project,http://www.mono-project.com/Olive

[100] Mono: Moonlight Project, http://www.mono-project.com/Moonlight

[101] Nigel McFarlane. Create Web applets with Mozilla and XML, 2003. http://www.ibm.com/developerworks/web/library/wa-appmozx/.

[102] Extensible Interface Markup Language, a universal language for user interfaces, http://www.ximl.org