



**MyUI: Mainstreaming Accessibility through
Synergistic User Modelling and Adaptability**

FP7-ICT-2009-4-248606

**Adaptation concept and Multimodal User Interface Patterns
Repository**

Public Document

Deliverable number	D2.2	Date of delivery	12-2011
Status	Final	Type	Report
Workpackage	WP 2 - Self-adaptive Multimodal User Interfaces		
Authors	FHG (Matthias Peissner, Dagmar Häbe, Andreas Schuller)		
Keywords	Adaptive user interface; multimodal user interface; adaptation engine; design patterns		
Abstract	This deliverable describes the approach to adaptive user interfaces as developed in the MyUI project. A conceptual framework for adaptive user interfaces is presented. This includes an Abstract Application Interaction Model (AAIM), a process and a technical infrastructure for user interface generation and adaptation, and a design pattern repository which serves as the basis for the modular approach to individualized and dynamically adapting user interfaces.		

© 2010-2012 MyUI Consortium

Table of Contents

EXECUTIVE SUMMARY.....	4
1. INTRODUCTION.....	5
1.1 THIS DOCUMENT.....	5
1.2 THE TASK IN RELATION TO THE MYUI PROJECT.....	5
2. OBJECTIVES AND REQUIREMENTS OF ADAPTIVE USER INTERFACES FOR IMPROVED ACCESSIBILITY	7
2.1 ADAPTIVE USER INTERFACES FOR ACCESSIBILITY.....	7
2.2 REQUIREMENTS ON ADAPTIVE USER INTERFACES TO IMPROVE ACCESSIBILITY.....	8
3. CONCEPTUAL FRAMEWORK FOR ADAPTIVE USER INTERFACES IN MYUI... 10	
3.1 ADAPTATIONS IN HUMAN-TO-HUMAN COMMUNICATION AS A MODEL FOR MYUI	10
3.2 SELF-LEARNING AND SELF-ADAPTIVE USER INTERFACES.....	10
3.3 ENHANCED USER PROFILING.....	11
3.4 DESIGN PATTERNS AS MODULAR BUILDING BLOCKS FOR ADAPTIVE USER INTERFACES ..	11
3.5 THREE-STAGE PROCESS OF USER INTERFACE GENERATION AND ADAPTATION	12
3.6 MODEL-VIEW-CONTROLLER AS CONCEPTUAL BASIS FOR THE MYUI ADAPTATION ARCHITECTURE	13
3.7 CONSISTENCY, TRANSPARENCY AND USER CONTROL.....	14
4. ABSTRACT APPLICATION INTERACTION MODEL (AAIM).....	15
4.1 PURPOSE OF THE AAIM.....	15
4.2 STATE MACHINE DIAGRAMS FOR ABSTRACT USER INTERFACE MODELLING	16
4.2.1 States and their interaction situations.....	17
4.2.2 States and their syntax.....	17
4.2.3 Interaction situations as basis for adapting the user interface	20
4.2.4 Interaction situations in states and transitions.....	20
4.2.5 Interaction situations and their syntax.....	21
4.2.6 References to functions.....	23
4.3 SOURCE CODE GENERATION FROM THE AAIM.....	23
4.4 EXAMPLE: AAIM FOR THE MYUI EMAIL APPLICATION	24
5. MYUI DESIGN PATTERNS FOR ADAPTIVE USER INTERFACES	27
5.1 DESIGN PATTERNS AS BUILDING BLOCKS OF A MODULAR USER INTERFACES	27
5.2 TYPES OF DESIGN PATTERNS IN THE MYUI DESIGN PATTERNS REPOSITORY	27
5.3 MYUI DESIGN PATTERN LANGUAGE.....	29
5.3.1 Relation substitutes.....	29
5.3.2 Relation requires	30
5.3.3 Relation is required by.....	30
5.3.4 Relation sets <variables> as required by	31

5.3.5	<i>Relation requires <variables> as set by</i>	31
5.3.6	<i>Relation sets <variable(s)> as used by</i>	32
5.3.7	<i>Relation uses <variable(s)> as set by</i>	32
5.4	MYUI DESIGN PATTERNS REPOSITORY	33
6.	USER INTERFACE PARAMETERIZATION	35
6.1	DEVICE MANAGER	36
6.2	DEVICE PROFILE	36
6.3	DEVICE-SPECIFIC PATTERNS	38
6.4	USER PROFILE	41
6.5	USER INTERFACE PROFILE	42
6.6	INDIVIDUALIZATION PATTERNS	43
6.7	CUSTOMIZATION PROFILE	46
7.	USER INTERFACE PREPARATION	47
7.1	INTERACTION PATTERNS	48
7.2	USER INTERFACE ELEMENTS	51
8.	USER INTERFACE GENERATION AND ADAPTATION	52
8.1	USER INTERFACE GENERATION	53
8.2	FEEDBACK TO THE MYUI USER AND CONTEXT MANAGEMENT INFRASTRUCTURE	53
8.3	ADAPTING THE USER INTERFACE DURING USE	54
8.4	ADAPTATION RENDERING PATTERNS	54
8.5	ADAPTATION DIALOGUE PATTERNS	55
8.5.1	<i>Explicit Confirmation before Adaptation</i>	56
8.5.2	<i>Explicit Confirmation after Adaptation</i>	57
8.5.3	<i>Automatic Adaptation with Implicit Confirmation</i>	58
8.5.4	<i>Automatic Adaptation without Adaptation Dialogue</i>	59
8.5.5	<i>Selecting the most appropriate system-initiated adaptation dialogue pattern</i>	60
8.5.6	<i>User-initiated Adaptation via User Profile</i>	61
8.5.7	<i>User-initiated Adaptation via User Interface Profile</i>	61
	REFERENCES	63

Executive Summary

This document is MyUI Deliverable D2.2 “Adaptation concept and Multimodal User Interface Patterns Repository”. It describes the innovative approach to adaptive user interfaces as developed in the MyUI project. The MyUI framework and infrastructure for adaptive user interfaces aim at increasing the usability and accessibility of ICT products for people with special needs by providing dynamic and highly individualized user interfaces.

Based on a summary of objectives and requirements of adaptive user interfaces for accessibility, a conceptual framework for user interface adaptation is presented. The MyUI adaptation concept builds on self-learning user profiling mechanisms to avoid tedious user enrolment and configuration dialogues (cf. D1.1). Self-adaptations during use are essential for the MyUI concept in order to cover individual development paths related to health and age. The MyUI concept considers dynamic user interface changes also as a potential source of disorientation, loss of control and as a significant risk for usability and user acceptance. Therefore, mechanisms to ensure best transparency and controllability play a key role in the MyUI adaptation concept.

Besides a general refinement of the adaptation concept, the main advances of this document – compared to the interim draft version (R2.2) - include the specification of Adaptation Patterns to ensure high level controllability and transparency for the end users and the development of an Abstract Application Interaction Model.

The Abstract Application Interaction Model (AAIM) defines the interaction between the end user and the application in a way which is independent of a specific appearance and concrete interaction mechanisms on the user interface. In an AAIM, the designer or developer of a MyUI application specifies application states, transitions and relevant interaction situations in a UML Statecharts-conform format. Then the AAIM serves as the starting point of automatically generating and adapting individual MyUI user interfaces. Therefore, it can be considered as the essential interface between application developers and MyUI adaptive user interfaces.

The MyUI process of user interface generation and adaptation is sub-divided into three steps:

(1) User Interface Parameterization

First, general user interface settings are defined to cover individual user needs and current context conditions. This first step can be regarded as “translating” information about the user (as drawn from the user profile) and the used device (as drawn from the device profile) to settings of the user interface and storing them in the User Interface Profile (UIP).

(2) Selection of suitable user interface components

On the basis of the current interaction situation and UIP, suitable display and control elements are selected for an optimal support of the individual user in her current interaction step.

(3) User Interface Generation and Adaptation

The selected user interface components are composed to an individual user interface. Decisions about necessary adaptations to the currently displayed user interface are made and executed.

These three steps are executed repeatedly in order to refine the individualized user interface in an iterative manner during the interaction. In a permanent process, relevant events from the user interaction are fed back to the User and Context Management Infrastructure in order to keep the user profile up-to-date. Every user profile update triggers the three-stage adaptation process again.

A design patterns language serves as a basis for a modular approach with re-usable software components as the building blocks for individualized and dynamically adapting user interfaces. The MyUI Design Patterns Repository includes different types of pattern with dedicated functions in the MyUI adaptation process. The structure and purpose of the patterns language and the different pattern types described in detail. Selected design patterns are included as examples in this report. However, for a full set of the MyUI design patterns the reader is referred to the online Design Patterns Repository at <http://myuipatterns.clevercherry.com>.

1. Introduction

1.1 This Document

This is public report D2.2 “Adaptation concept and Multimodal User Interface Patterns Repository” is a major deliverable of the MyUI project (ICT-248606).

The results are specifically the outcome of Task 2.2 “Concepts for self-adaptive multimodal user interfaces” and Task 2.3 “Multimodal User Interface Patterns Repository”.

1.2 The task in relation to the MyUI project

The conceptualisation and implementation of adaptive user interfaces for users with special needs are major tasks in the MyUI project. The effectiveness and quality of the adaptive user interface solutions are essential for the success of the entire project. Therefore, this public report D2.2 is a core deliverable of the project.

Important input to this task and internal report stems from the following project activities and documents:

Task 2.1 with Deliverable D2.1 “Requirements for User Interface Adaptation”:

- General requirements of user interfaces for the MyUI target user groups
- Requirements and state-of the art of adaptive user interfaces for accessibility
- Important user impairments as a basis for a pragmatic user profiling approach

Tasks 1.1 and 1.2 with Deliverable D1.1 “Context ontology, user modelling concept and context management architecture”:

- User model ontology and user profiling mechanisms

WP 4 with Deliverable D4.1 “Demonstration scenarios”:

- Description of main scenarios of use and personas as a basis for pattern development

Adaptive systems can be considered to consist of three main levels (Weibelzahl, 2003; summarising Oppermann, 1994 and Jameson, 2001):

- Afference – collection of observational data about the user.
- Inference – creating or updating a user profile based on that data.
- Efference – deciding how to adapt the system behaviour.

The work in WP 2 concentrates on the efference functions of the MyUI adaptive system. Afference and Inference are covered by WP 1 and described in Deliverable D1.1. This means, that the modules developed in WP 2 and described in this document assume to start from a somehow reliable user profile. The mechanisms which collect and interpret information about the user and his context are not focused in this document but subject of D1.1 (see Figure 1).

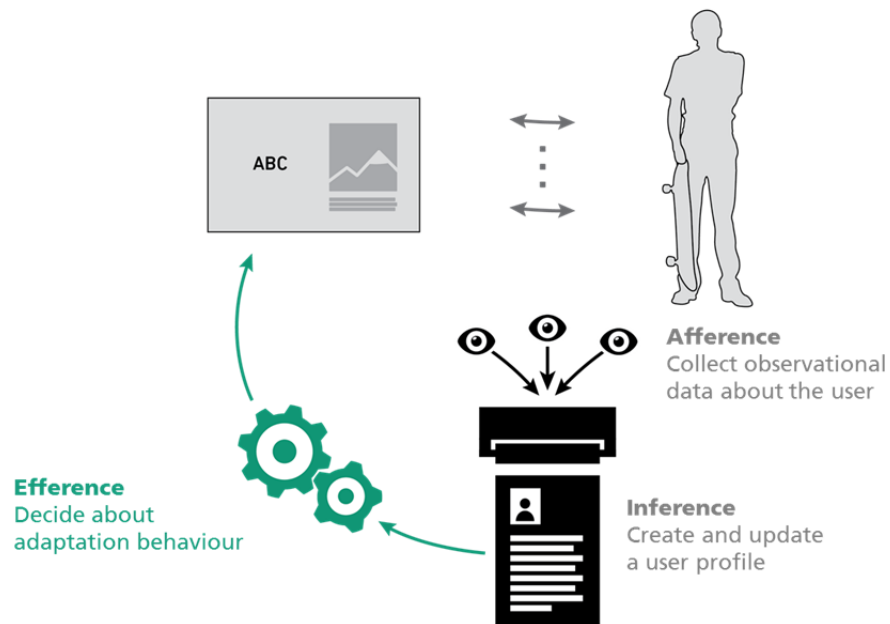


Figure 1 This deliverable focuses on the efference functions of the MyUI adaptive system

The main challenge of the research work summarized in this report is the conceptualisation and implementation of an approach to generating and adapting user interfaces during run-time which allow end users with special needs to interact with a system (here an iTV device and services) in a comfortable manner.

2. Objectives and requirements of adaptive user interfaces for improved accessibility

2.1 Adaptive user interfaces for accessibility

Adaptive user interface technologies have a long tradition (see e.g. . Nielsen, 1993) but still are envisioned to yield great advantages for the usability and accessibility of advanced technical systems (cf. Harper, Rodden, Rogers, & Sellen, 2008), especially for people with special needs. Modern recognition, sensor and agent technologies aim at interpreting user actions and contextual information in terms of user goals and intentions and providing services as an adequate system reaction. Adaptive user interfaces working on user and context models are widely considered as empowering the user to benefit from system intelligence and to control complexity.

However, most of the published approaches to adaptive user interfaces aim at different purposes than accessibility for users with special needs. Early systems as the Doppelgänger (cf. Orwant, 1994) and the Lifestyle Finder (cf. Krulwich, 1997) were built in order to present or suggest interesting content to a user with specific interests. In their recent book, Jannach et al. (Jannach, Zanker, Felfernig, & Friedrich, 2010) provide an extensive overview of personalized recommendation systems in diverse application fields. Similarly, adaptive e-learning systems such as ELM-ART II (cf. Weber, & Specht, 1997) aim at promoting efficient learning by adapting the instruction material to the learner's individual progress and cognitive level.

Another category of adaptive user interfaces support the users by providing appropriate levels of assistance and guidance in order to improve the usability and comfort of use, e.g. "Augur" (Hartmann., Schreiber, & Mühlhäuser, 2009), "Flexcel" (Thorrtas & Krogsoeter, 1993) and "SmartCal" (Krzywicki, Wobcke, & Wong, 2010).

Modern Design-for-All approaches discuss flexible user interfaces which are generated on the basis of a specific user profile, as one common user interface for all often fails to meet many users' special needs. In the EU-project ASK-IT (IST-2003-511298: Ambient Intelligence System of Agents for Knowledge-based and Integrated Services for Mobility Impaired users), a user taxonomy has been developed which can be used as a starting point to describe specific needs (cf. Ringbauer, Peissner, & Gemou, 2007). The follow-up project OASIS has developed an adaptation framework which relies particularly on a library of adaptive widgets (cf. Leuteritz, Widlroither, Mourouzis, Panou, Antona, & Leonidis, 2009). In the research project SUPPLE at the University of Washington, a UI toolkit has been developed for building applications with automatically generated and personalizable user interfaces. Main research and development activities of the SUPPLE project include the adaptation to a person's device (Gajos, & Weld, 2004) and physical disabilities (Gajos, Wobbrock, & Weld, 2008).

According to Brusilovsky and Maybury (2002), user interface adaptations can take place in three areas of a system: (1) the selection of content to be displayed or recommended to the user, (2) the presentation of information including colours, font sizes, layout, etc., and (3) navigation which defines the possible paths users can take through an application in order to access a certain information or functionality. Their focus on adaptive web applications might explain why they do not mention adaptations in interactions mechanisms such as supporting different input devices and modalities. In an article about an adaptive email system, Peng and Silver (2007) distinguish (1) content adaptations (e.g. prioritizing of relevant content elements), (2) adaptations of the presentation format (e.g. text-graphics ratio) and (3) media selection (e.g. audio instead of video), and (4) adaptations to specific devices in order to cover device-specific output and input restrictions. However, it should be recognized that presentation format and media selection can be regarded as one common aspect and adaptations to a specific device mostly concern one or more of the other already mentioned adaptation areas.

In summary, user interface adaptation can apply to the following four design areas: (1) adaptive content selection; (2) adaptive presentation; (3) adaptive navigation; and (4) adaptive input

mechanisms including the support of different input devices and technologies (cf. Figure 2).

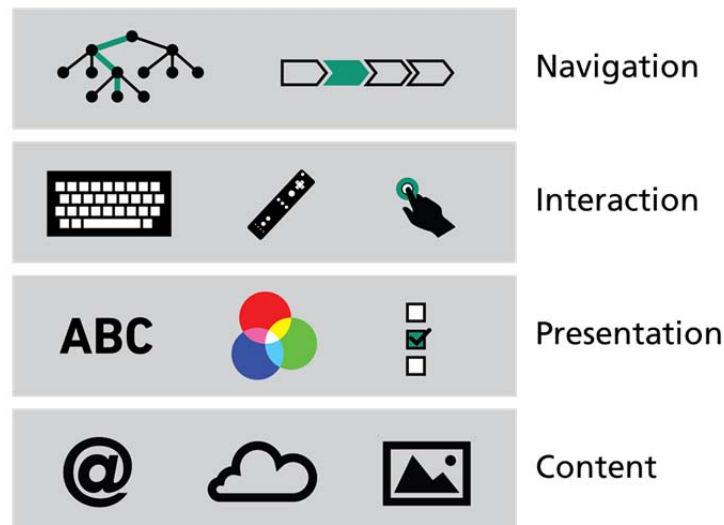


Figure 2 Adaptions in user interfaces can apply to the displayed content, the presentation mode, the interaction mechanisms and navigation paths

In many cases, user interface adaptations concentrate on very specific areas of a user interface whereas other design areas are kept stable. In personalized e-commerce applications, for example, only the selection of displayed content elements is adapted to fit the user's assumed preferences. Adaptive user interfaces which aim at improving the accessibility, however, will have to take into consideration all design areas – except content adaptations which are not specific for accessibility. The great diversity in perceptual, motor and physical capabilities and in contextual conditions requires adaptations of presentation, interaction mechanisms and navigation.

Moreover, there are significant interaction effects between adaptations in the different design areas. Adaptive mechanisms designed in order to improve one aspect of interaction will often lead to increased effort or even problems in another dimension (cf. Findlater & Gajos, 2009). For example, when font sizes are increased in order to address vision impairments, selection lists might need to be split to be displayed on two screens, which increases attention and memory demands on the user. Thus, developing user interfaces that support accessibility by providing extensive adaptivity is a highly complex challenge which appears to be not yet mastered.

MyUI aims at providing individualized user interfaces which are accessible to a broad range of end users by adapting presentation, interaction and navigation aspects of the user interface. Besides the above described challenge of potential conflicts between adaptations to compensate different special user needs, the MyUI adaptation approach addresses a number of requirements as described in the next section.

2.2 Requirements on adaptive user interfaces to improve accessibility

As pointed out by MyUI Deliverable D2.1, adaptive user interfaces can help to support accessibility. From a user perspective, the design of effective and acceptable adaptations, however, is not straightforward. A basic challenge is that “the benefits of correct adaptations must outweigh the costs, or usability side effects, of incorrect adaptations” (Findlater & Gajos, 2009). This demand can be understood in manifold ways, as, for example, adaptive systems are often criticized for their lack in consistency and therefore dependability and for their autonomous adaptation behaviour which can result in a loss of user control. Major challenges in the design of concrete adaptive user interface solutions include creating design solutions that fit the individual users' needs and finding adaptation mechanisms that lead to understandable and trustworthy system

behaviour. Moreover, the basic principles of adaptation play an important role in deciding upon the effectiveness and success of adaptive user interfaces. For the design of the conceptual framework of MyUI adaptive user interfaces, we focus particularly on the following major requirements:

- **Adapting presentation, interaction and navigation**
The MyUI project addresses accessibility problems typically associated with aging and stroke in quite a broad sense. Therefore, MyUI user profiles cover perceptual, cognitive, and motor characteristics and impairments of individual users. As a consequence, all of the three above mentioned user interface domains presentation, interaction and navigation must be subject to adaptation.
- **Manifold user interface solutions: modular and extensible**
Individualized user interfaces for heterogeneous user groups require a just as broad spectrum of user interface solutions. A modular approach will be needed to manage a huge amount of possible user profiles and respective user interface solutions. For practical reasons, extensibility of the modular approach will be important in order to support a quick start with a manageable subset of design solutions and later extensions.
- **Consistent and sensible user interfaces**
Adapting user interfaces to multi-dimensional user profiles can be viewed as resolving a multidimensional problem where conflicts and inconsistencies can easily occur. Effective rules and mechanisms for the adequate selection and composition of design solutions throughout an entire application will be fundamental for achieving consistent and sensible user interfaces.
- **Self-learning and adapting during use**
Intelligent user interface adaptation mechanisms avoid tedious configuration procedures. Not only before but during the interaction, the system is collecting information about a specific user which is stored in a user profile as basis for automatic user interface individualizations. This self-learning process leads to dynamic changes of an individual user profile during the interaction that will be reflected in a dynamically adapting user interface. Furthermore, aging and recovering from a stroke are often associated with significantly altering capabilities which will lead to different needs and changed user profiles over time. Therefore, the adaptation framework must support run-time rendering and run-time adaptations of the user interface (cf. “adaptation during use” as described by Dieterich et al., 1993). The design of transitions from one instance of a user interface to another becomes an important design issue.
- **Transparent and controllable**
System-initiated user interface adaptations can lead to loss of orientation and loss of control from the end user perspective. In order to assure high levels of usability and user acceptance, the MyUi approach needs to provide mechanisms which help the end users to recognize and understand changes of the user interface. Moreover, it will be necessary to establish an optimal balance between automatic adaptations and explicit end user control.

3. Conceptual Framework for Adaptive User Interfaces in MyUI

The following sub-sections describe the general principles and concepts of user interface adaptation in MyUI. This section provides an overview of the core concepts. Most of the presented concepts are described in more detail in other dedicated sections of this deliverable.

3.1 Adaptations in human-to-human communication as a model for MyUI

The model for user interface adaptations in the MyUI project is the gradual adaptation of human communication behaviour to a specific communication partner. In the beginning, both partners show quite careful and neutral communication behaviour. Over time, they learn to know each other better and better. This mutual understanding is established and improved by perceiving the partner's feedback to one's own communication acts, e.g. explicit remarks, emotional facial expressions, mimics and gestures, etc. This increasing understanding of the partner helps to accommodate his own behaviour to the individual traits of a specific partner. Without the need of an explicit interview which could capture personal preferences and attitudes, human communicators adapt to each other in a quick, natural and smooth way during the interaction.

MyUI strives to imitate such a smooth and natural adaptation process in accessible and highly individualized user interfaces.

3.2 Self-learning and self-adaptive user interfaces

MyUI aims at minimizing the need for an initial user interface configuration or user enrolment. MyUI systems are delivered in a “raw” state that have the potential of evolving diverse concrete shapes in combination with a specific end user and her/his surroundings. Then, the system is learning to more accurately adapt to a specific end user and relevant contexts or situations. This adaptation requires user and context information which is shared across all adaptive applications in the environment. The gained synergies result in a network of personal applications which can learn from each other on-the-fly and act in coordination. Figure 3 illustrates the closed loop of interaction, sensing and interpreting information about the user and the interaction environment, user profiling and composing an individual user interface on the basis of the currently available user profile.

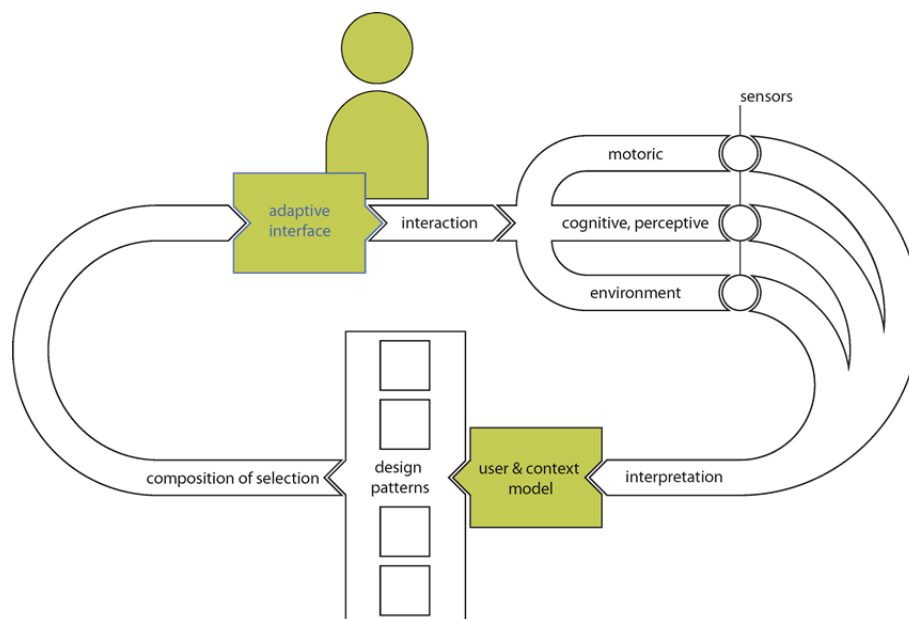


Figure 3 Self-learning and self-adaptive user interfaces in MyUI

This MyUI adaptation model is consistent with the high-level model of adaptation in adaptive user interfaces as described by Paramythis, Totter, & Stephanidis (2001) which is based on Totterdell's two-level architecture of adaptation (cf. Totterdell & Rautenbach, 1990).

Most information relevant for the MyUI user profile will be extracted from the end user's interaction behaviour with the current user interface. Examples for informative user interface events include time-outs (the user does not react to a system prompt within a certain time frame), repeated undos (the user seems to repeatedly select wrong options by accident) and detours. Hardware sensors as, for example, eye tracking in order to capture the user's attention, are also involved in the MyUI user profiling process, but due to their accessibility limitations software sensors are preferred. The mechanisms for sensing and interpreting user profile-relevant information are described in more detail in deliverable D 1.2.

User interface adaptations in MyUI follow a three-stage process to allow for adaptations during use while providing high-level consistency throughout the entire user interface. The MyUI adaptation process is described in sections 6, 7 and 8.

3.3 Enhanced user profiling

In addition to self-learning user profiling, an initial user enrolment dialogue is foreseen which is not mandatory but can help users to create a first instance of a reliable user profile in a playful manner in order to take full advantage of the MyUI system from the very beginning.

Additionally, the MyUI platform offers a couple of games which track the user performance while playing and use this information as an input to user profile refinements. These games challenge user capabilities which are relevant for the interaction success with modern ICT products such as the MyUI services. For example, specific games for attention, vision and motor skills provide relevant information for the user profile. On the other hand, these games are designed to motivate and reward users for repeated playing and individual improvements. Thereby, the games support rehabilitation and overall fitness of the target user groups.

Finally, end users will always have access to their personal data. MyUI user interfaces permanently provide opportunities to create, view, and modify user profile data and user interface settings. This information explicitly provided by the end user is fed back to the user profiling mechanisms as described in more detail in the relevant deliverables of work package 1.

3.4 Design Patterns as modular building blocks for adaptive user interfaces

MyUI follows a modular approach to user interface generation and adaptation which relies on the composition of multimodal user interface patterns. Each pattern is related to specific end user characteristics and context requirements which are explicitly addressed by the design. Thus, individual accessibility is achieved by combining design patterns that suit for a certain end user and context of use.

MyUI design patterns cover all above mentioned user interface adaptations of presentation, interaction and navigation. According to Borchers' classification of patterns in terms of their function, patterns can be classified into those that address mainly issues of interface output (*perception*), and those that deal with interface input (*manipulation*) or *navigation* through the system (cf. Borchers, 2000). The MyUI patterns repository covers all these functions, too. But it uses a different taxonomy of patterns. MyUI patterns are categorized in terms of their functions in the adaptation process. In order to assure high consistency throughout the user interface and to avoid conflicts in the adaptation process, different patterns work on different levels and different aspects of the user interface. A detailed description of the different pattern types and their roles can be found in section 5.2.

3.5 Three-stage process of user interface generation and adaptation

MyUI adaptive user interfaces are generated and adapted in a three-stage process (see Figure 4):

1. *User Interface Parameterization*

In a first step, general user interface characteristics are set in order to adapt the overall user interface appearance and interaction mechanisms to a specific user, specific I/O devices and a desired corporate (or project or product) identity. These general user interface settings are done via global variables and stored in the MyUI User Interface Profile. In the following, this first step is referred to as *User Interface Parameterization*. A MyUI User Interface Profile is initiated at the beginning of a new interaction session with a MyUI application. A repeated User Interface Parameterization cycle (i.e. user interface profile update) is triggered when newly available information about the user and his/her context leads to a significant change in the user profile. User Interface Parameterization is described in detail in section 6.

2. *User Interface Preparation (Component selection)*

On the basis of the current interaction situation and relevant variables of the user interface profile the most suitable display and control elements are selected from a repository of user interface components and elements. This second step makes sure that the selected user interface building blocks correspond to individual user needs and device-related requirements. It is triggered whenever a new state of the application is entered (i.e. a new interaction situation is active) and after each user interface profile update. The new selection of display and control elements is compared with the currently displayed selection. If a difference between the current and the new components is identified, adaptation patterns to switch from the current to the new user interface are selected. This second sub-process is called *User Interface Preparation* and is described in section 7.

3. *User Interface Generation and Adaptation*

The selected display and control elements are composed and rendered to an individualized user interface. This rendering process takes place at the beginning of a new interaction session with a MyUI application in order to generate the interface. The same process is also triggered when in the user interface preparation process (step 2) a new set of display and control elements is selected. In this case, also the selected adaptation patterns are executed to switch from the current to the new user interface. This third step in the MyUI adaptation process is called *User Interface Generation and Adaptation* and is described in detail in section 8.

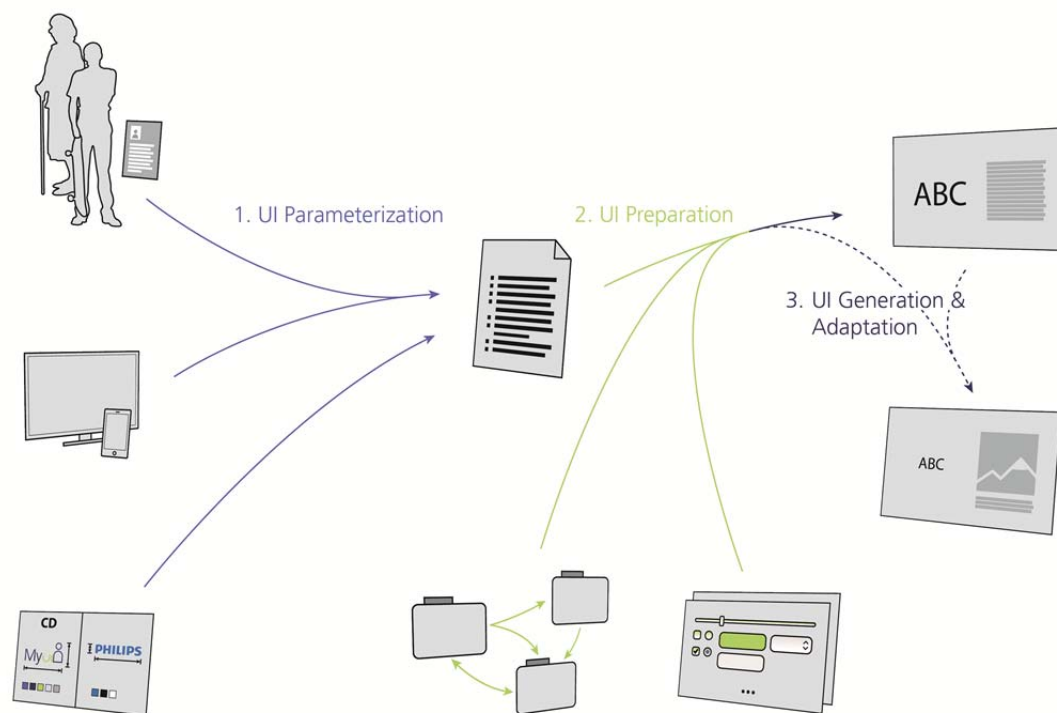


Figure 4 Three-stage MyUI user interface generation and adaptation process

During the interaction between the end user and the MyUI application permanent run-time processes sense relevant events from the interaction and provide feedback to the User and Context Management Infrastructure to refine the user profile. Examples for sensed interaction events include time-outs, rejection or acceptance of interface adaptations and explicit user interface profile changes initiated by the user. After significant user profile updates the above described process of user interface adaptation starts again.

3.6 Model-View-Controller as conceptual basis for the MyUI adaptation architecture

Model-View-Controller (MVC) is a software architecture paradigm that separates the application's classes and objects into three categories: **Model**, **View** and **Controller**. The model represents and maintains the underlying dataset (e.g. entities from a database) of an application. The view consists of interaction and presentation components which are displayed on the user's output device. The view displays data from the model. The controller reacts to the user input. It handles the appropriate interaction with the user and executes specific application functions. Thereby the controller is responsible for manipulating the current data set of the model and for updating the view (cf. Gamma, Helm, Johnson & Vlissides 1994). The dependencies of the different MVC-units are illustrated in Figure 5.

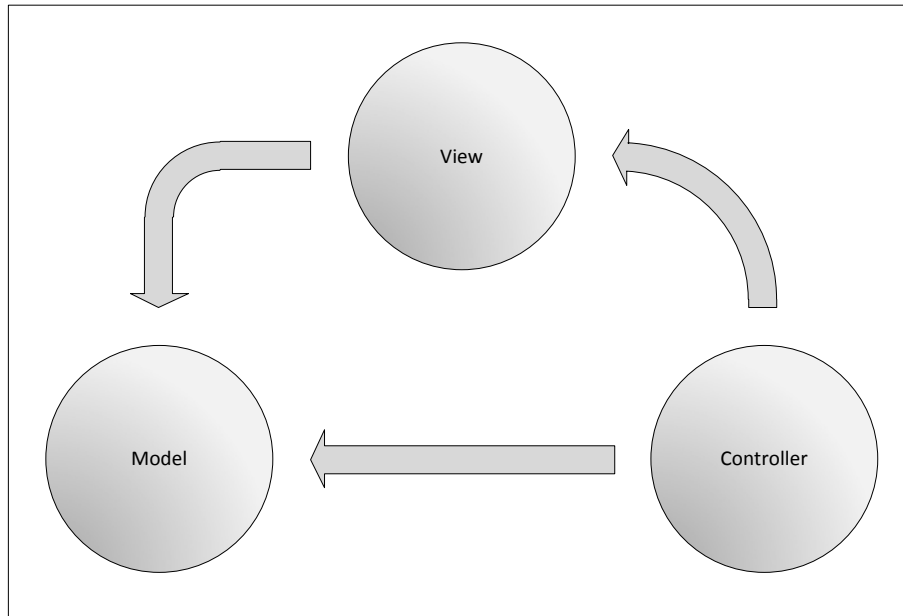


Figure 5 Dependencies in the Model View Controller paradigm

The separation into these three components facilitates a better decoupling of the application's objects which makes the development more flexible and reusable. By establishing a Publishing / Subscriber behaviour in MVC the dependency between the model and the view will be reduced. For instance, every view of an application has to ensure that it displays the current value of the model, as soon as it has been changed by the controller. For that reason every view that subscribes to a specific model will be immediately notified after the model's data has been changed.

The Model-View-Controller plays an important role in the framework for developing adaptive user interfaces in MyUI. When developing adaptive user interfaces it is particularly important to build on a clear division of the different responsibilities and dependencies of the individual components. Both, the MyUI adaptation framework core and the Abstract Application Interaction Model (described in section 4) implement MVC as a conceptual basis for the MyUI adaptation architecture. The MyUI adaptation engine (controller) is responsible for transforming a user profile into a user interface profile (model). If any changes occur to the user interface profile, all currently displayed screens (views) of a MyUI application will be notified to adapt the user interface.

3.7 Consistency, Transparency and User Control

Self-learning and self-adaptive user interfaces offer great opportunities for accessibility and improved usability for a broad range of diverse users. However, they also pose significant challenges in terms of usability, trustworthiness and acceptability.

Consistency is a main quality of usable and reliable user interfaces. System-initiated adaptation will lead to changes in the appearance of the user interface. In many cases, these adaptations will occur in situations of interaction problems. Therefore, it is important that changes triggered by adaptation processes do not lead to further disorientation but support the user in recovering from an experienced interaction problem.

A second very important topic for user interface adaptations is their transparency. The users must be able to recognize changes in the user interface and they should also be able to understand why the change has been made.

Finally, intelligent and automatic adaptations must not lead to situations where the end user feels like losing control over the user interface. It is highly important to offer effective means of keeping

full user control over the interface and potential changes. Examples include rejecting and undoing system-initiated adaptations and user-initiated user interface changes.

MyUI recognizes that the design of adaptation processes is a major design issue for the usability and acceptability of adaptive user interfaces. In order to address this important field, a dedicated category of MyUI design patterns has been developed: the Adaptation Patterns. They inform the end users about intended adaptations, and provide the end users with effective control mechanisms.

4. Abstract Application Interaction Model (AAIM)

MyUI user interfaces can have very different appearances. The MyUI design patterns repository allows for a huge variety of individual user interfaces by providing modular building block which can be combined in manifold ways. Despite their differences in presentation, interaction and navigation aspects, all user interface variants of one specific MyUI application provide the same functionality to the end user and follow the same application logic.

The three-stage MyUI user interface generation and adaptation process is an automatic process. This changes the application developers' role significantly. Their impact on the specific user interface appearance is minimized in favour of an automated and rule-based user interface generation process which includes the provision of different user interfaces for users with different needs. The developers' main responsibility in MyUI is defining the functionality and application logic in an *Abstract Application Interaction Model* (AAIM). The AAIM describes the interaction between the user and the application in a way which is independent of a specific appearance and concrete interaction mechanisms. It serves as a basis for the generated and adapted user interfaces by defining the common ground of all possible user interfaces.

4.1 Purpose of the AAIM

The AAIM fulfils the following important functions in the MyUI framework (see Figure 6):

- **Interface between developer and the MyUi framework**
The AAIM is the only necessary artefact to be provided by the application developer in the course of developing an adaptive MyUI application. Thus, the AAIM represents the primary point of contact between the application developer and the MyUI adaptation framework with its design patterns repository. As a consequence, creating an AAIM should be made as easy as possible for potential industrial MyUI users in order to support a wide-spread use of MyUi technologies in the industry.
- **Basis for user interface generation and adaptation**
The AAIM is the starting point of the MyUI user interface generation process. For each state of a MyUI application, it defines which interactions situation are presented to the user. Interaction situations represent the interaction options a user is provided at a certain point in the application, e.g. viewing a list of items, triggering certain functions, going to another place in the application, etc. Via Interaction Design Patterns of the MyUI Patterns Repository, each interaction situation is associated with a bundle of user interface components suitable for the respective situation. Therefore, the AAIM is the essential basis for the adaptation sub-process in which the most suitable user interface components and elements are selected (cf. User Interface Preparation).
- **Interface between user interface and application functions**
The AAIM specifies also application functions to be triggered when a state is entered or in association with a specific transition from one application state to another. Application functions can set certain system variables, manipulate the application data base or perform transactions as, for example, send an email. Thus, the AAIM establishes relations between the user interface (view in MVC) and application functions to be performed in the back end of the system (controller in MVC).

– **Interface between user interface and application databases**

Via interaction situations (and related interaction design patterns), the AAIM specifies which user interface components to be selected for presentation at the user interface. Most user interface components will need application content, e.g. a selection element requires items from the application database from which the user can select. The AAIM defines relations between interaction situations and application databases, i.e. it connects a specific interaction situation with a data base from where the selected user interface component gets its application content. Thus, the AAIM establishes relations between the user interface (view in MVC) and the application data base (model in MVC) via data acquisition functions (controller in MVC).

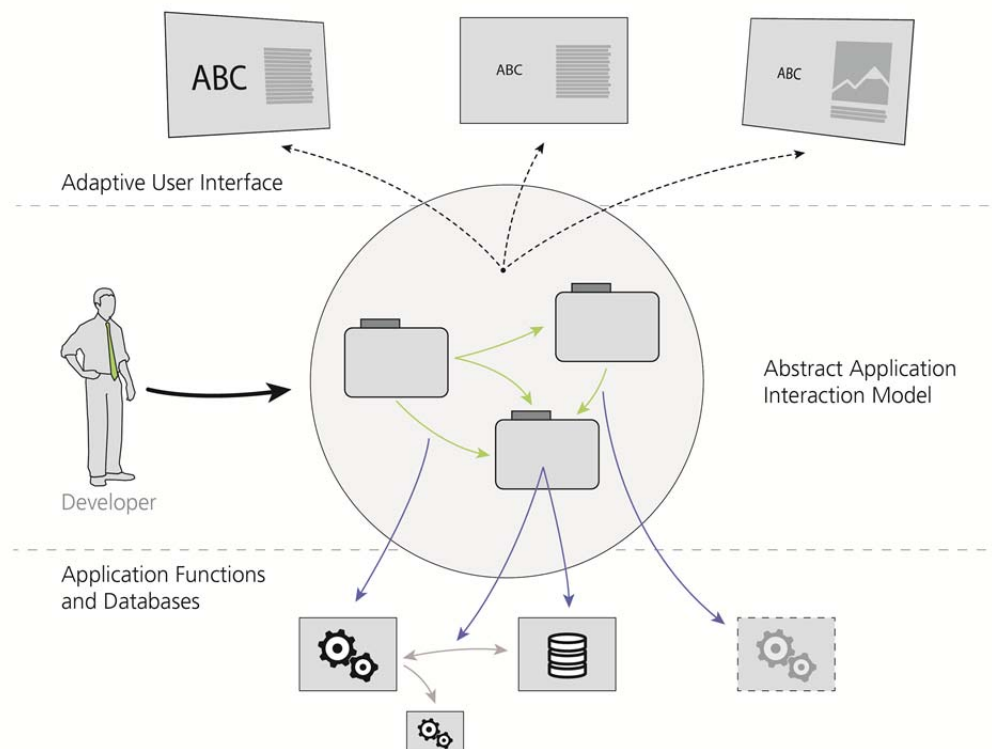


Figure 6 Role of the Abstract Application Interaction Model (AAIM) in the MyUI framework

4.2 State Machine Diagrams for abstract user interface modelling

The abstract application interaction model (AAIM) in MyUI is based on and extends the UML 2 State Machine Diagram (cf. OMG, 2010 or Jeckle et al., 2004). In MyUI, statecharts have been identified as the best suitable modelling format for the purpose of specifying the interaction with adaptive user interfaces. The joint VUMS cluster deliverable VERITAS D1.6.4 provides an overview of current user interface modelling languages. In contrast to the approaches presented there, the statecharts-based approach in MyUI works on a higher abstraction level. Statecharts allow modelling the interaction without going into details about the presentation modalities or used control elements. All these aspects are subject to adaptations in MyUI and therefore not part of the MyUI AAIM.

The MyUI AAIM specifies the interaction between an end user and a MyUI application in the format of a statechart. Each single state describes a specific status of the application and the interaction options provided to the user. At any point of time the application is exactly in one and only one single state of the AAIM.

4.2.1 States and their interaction situations

Each single state is defined by its name and the included interaction situations (IS). States can include interaction situations of four different types:

1. **Primary IS (mandatory)**
The primary IS represents the main purpose of the current state. It can cover any IS supported by the interaction patterns of the MyUI patterns repository – except the below listed interaction situations *SelectFunction* and *SelectGoTo*. The primary IS and its main purpose are typically associated with a specific object which is presented at the user interface to be viewed or manipulated by the end user. This object is called the “main object” of the current state.
2. **IS *SelectFunction* (optional)**
The optional IS *SelectFunction* specifies a set of additional application functions which can be triggered by the user in the current single state. Typically, these application functions are applied to the main object of the current state and extend the functionality provided by the primary IS.
3. **IS *SelectGoTo* (optional)**
The IS *SelectGoTo* specifies a set of target states to which the user can navigate. In contrast to the IS *SelectFunction* which also often results in going to another state of the application, the *SelectGoTo* interaction situation triggers only state transitions and does not affect the application data.
4. **IS *MetaGoTo* (only for the main state of an application)**
The IS *MetaGoTo* is a special case of the IS *SelectGoTo* which can apply to the main state of an application. It provides generic navigation functions such as “go back to the previous state” and “go home” (to the main menu) in a consistent manner for an entire application.

4.2.2 States and their syntax

Figure 7 presents the notation of a single state of the AAIM. State names use upper camel case. In addition to interaction situations, states can also include the initialisation and setting of global variables to be used in the AAIM for a more flexible and concise description of the user interaction.

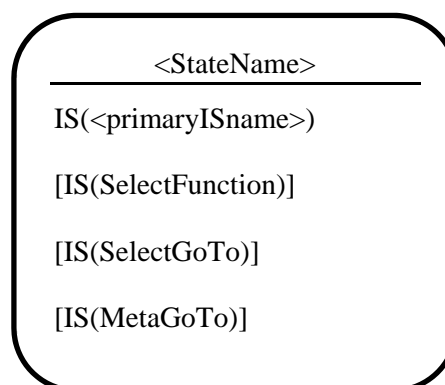


Figure 7 Notation of a single state of the AAIM

State transitions in the AAIM are defined by a starting and target state. Transitions where starting and target states are the same are called self-transitions. Transitions are triggered by events, typically user interactions. Together with a transition an IS-related activity can be triggered (see Figure 8).

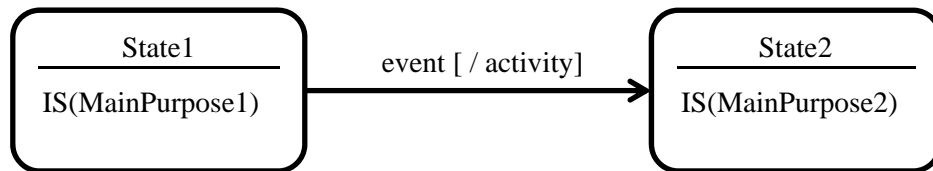


Figure 8 Notation of a state transition in the AAIM

Initial states are marked by a circle filled out in black. They do not have incoming transitions and do have exactly one outgoing transition to the first state of the AAIM as illustrated in Figure 9.

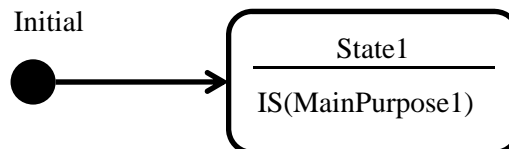


Figure 9 Notation of an initial state in the AAIM

Final states are marked by a smaller circle filled out in black that is surrounded by another circle. They do not have outgoing transitions and have at least one incoming transition as shown in Figure 10.

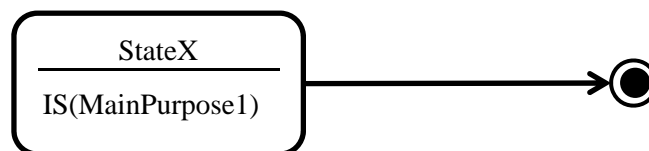


Figure 10 Notation of a final state in the AAIM

Several single states can be contained in one composite state to enable a more compact representation of the AAIM. Composite states in the AAIM are characterised by a name and the single sub-states they include. They do not include a primary interaction situation. But they can include any type of the optional interaction situations (*IS SelectFunction*, *IS SelectGoTo* and *IS MetaGoTo*) which have been presented at the beginning of this section (see Figure 11).

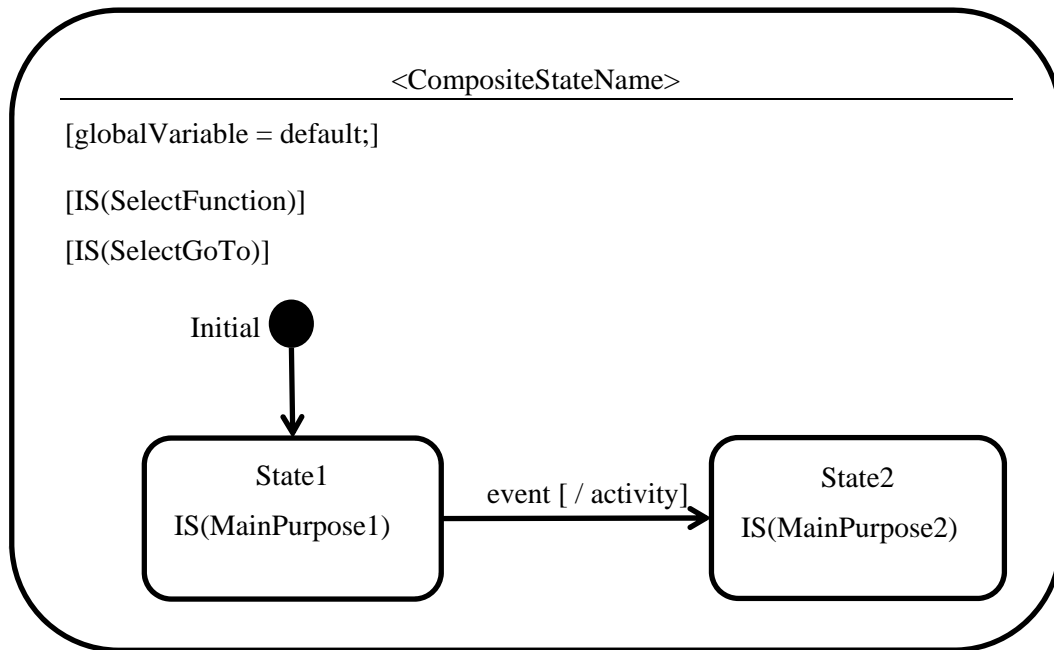


Figure 11 Notation of a composite state

When a composite state is entered, its initial sub-state is activated per default. All other sub-states of a composite state can be activated directly from outside the composite state by transitions which lead directly to the certain sub-state.

The specifications of a composite state apply to all sub-states it includes. Composite states can include composite states again. If the content of a composite state is too extensive to be presented in the AAIM chart, the body of the composite state, including the sub-states and other specifications, can be hidden in the overall AAIM chart and presented separately on a separate AAIM chart. Composite states which are detailed in a separate chart are marked with a glasses symbol (see Figure 12).

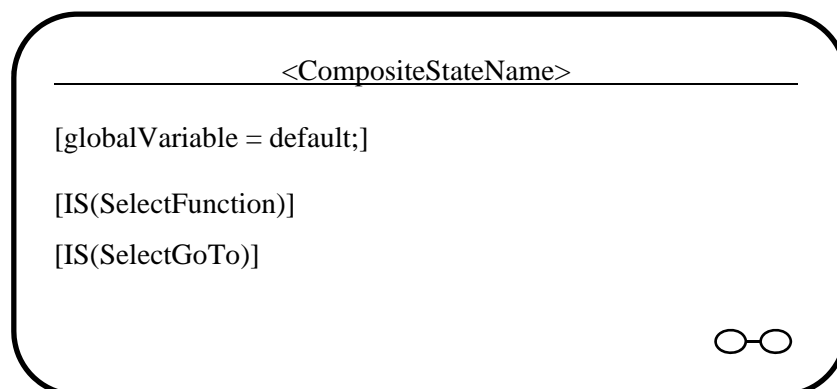


Figure 12 Notation of a composite state with hidden body content

Composite states with more than one region, each including nested disjoint states, are briefly mentioned here for completeness and would allow to model parallel processes within a certain state of the application (see OMG, 2010 for a detailed description). Even if the current reference email application for the interactive TV is described by a single process state model, other applications running on a different device might profit from composite states with regions when setting up their AAIM to sufficiently describe the full spectrum of interaction options available at a specific stage of the application provided by parallel processes.

In some cases, it is reasonable to return to the latest visited sub-state of a composite state when re-entering it. This can be done by means of a history mechanism which stores this information before leaving the composite state. The subdivision into deep and swallow history specifies the hierarchical level on which the composite state will be entered again (see OMG, 2010 for a detailed description).

4.2.3 Interaction situations as basis for adapting the user interface

The concept of interaction situations (IS) is essential for the MyUI adaptation framework. An interaction situation is an abstract super-set of user interface components and controls which serve the same interaction purpose. Interaction situations represent the interaction options a user has at a certain point in the application. Interaction options include all activities a user can perform at a given point in the application, e.g. perceiving information, providing specific input, selecting options, etc. Typically, more than one user interface component or element can be used in one certain interaction situation, e.g. selecting a single item from a set of items can be supported by a selection list, a drop-down list or even an audio menu. This flexibility is an essential basis for user interface adaptations in MyUI. Depending on individual end user needs, the most suitable control or display element can be selected for a given interaction situation. These different variants of user interface components for a given purpose are referred to as bundles of interaction situations (see section 5 for a detailed description of the MyUI patterns).

In order to clearly distinguish interaction situations from the closely related interaction patterns, dedicated naming conventions are used. Whereas interaction situations are named like user interface components and widgets (system perspective), interaction situations are named from a perspective of possible user actions (user perspective). Thus, the names of interaction situations always start with a verb to reflect a possible user action at the current point in the application.

As an outcome of the interaction with an end user, a user interface component (as provided by an interaction pattern) returns a specific interaction result. This interaction result is typically coded in the format of one or more variables to communicate the interaction outcome from the current interaction situation to following situations.

4.2.4 Interaction situations in states and transitions

Interaction situations are a substantial property of states in the MyUI AAIM. Every state includes one or more interaction situations (cf. section 4.2). Interaction situations apply to the state in which they appear. Interaction situations in a composite state refer to all sub-states of the composite state and persist as long as one of the sub-states is active. This can be used also for the definition of generic navigation options which are available throughout an entire application or even across different applications, e.g. home navigation and back navigation of the IS *MetaGoTo*.

An interaction situation of a state can be associated with respective transitions from the state to other states. An interaction situation, for example, which provides a set of functions from which the user can select in the current state, is always associated with a set of transitions from the state to other states in which the triggered functions are executed.

Interaction situations can appear in states (see section 4.2.1) and as arguments of transitions between states of the AAIM. Interaction situations at transitions are needed to retrieve confirmation or additional information from the end user needed to perform the transition and to enter the next state. As an example, some interaction situations *SelectFunction* require arguments for executing the selected function, e.g. the destination folder of the “move to folder” function or an explicit user confirmation before the execution of an irreversible action. These arguments are then collected by a separate interaction situation, e.g. a dialogue box which requests the needed information from the user.

4.2.5 Interaction situations and their syntax

In the following, further characteristics of the different types of interaction situations are described in detail. The notation of interaction situations is specified in the Backus-Naur-Form (BNF) consistent to BNF-conventions in the OMG Unified Modeling LanguageTM (OMG UML), Superstructure, Version 2.3, May 2010. The following rules are considered:

- non-terminals are enclosed by angle brackets;
- terminals are given in single quotes;
- definitions use the '::=' operator;
- repetitions (at least one) are marked by an asterisk;
- alternatives are separated by '|';
- optional items are enclosed by square brackets;
- a collection of terms is enclosed by round brackets.

4.2.5.1 Primary IS

A primary IS represents the main purpose of a single state. Each primary IS is characterised by its name and the content or data to be provided to the user via the activated interaction pattern. The content can be given directly in the AAIM state chart or it can be referred to by a data acquisition function which retrieves the application content from the data model in a predefined format. As an additional feature, the AAIM can include parameterization details for the displayed content, e.g. for prioritisation of the content.

Formally, the notation of a primary IS in a state is defined as follows:

```

<primary IS> ::= 'IS(' <primary IS name> ')', ' <data provider>
<data provider> ::= <direct content input>* | <data acquisition functions>
<data acquisition functions> ::= [ ( ' _ ' <function> ' , ' )* ] ( ' _ ' <function> )
<function> ::= <function name> ' ( ' [ <function parameters> ] ' ) '
<function parameters> ::= [ ( <parameter> ' , ' )* ] <parameter>

```

Primary interaction situations can be associated with specific interaction results which represent the outcome of the end user's interaction with the related user interface component (i.e. interaction pattern). This interaction result is usually represented as one or more variables, e.g. an entered text string or a selected option. These variables help to communicate the interaction results to other states which are entered in subsequent steps of the interaction.

Some primary IS are associated with an outgoing transition¹, e.g. selecting an item from a "list with attributes" directly leads to leaving the current state and moving to another state. In this case the transition label is formally described as follows:

```

<transition label> ::= <primary IS name> ':' <interaction result>
<interaction result> ::= [ ( <variable> ' , ' )* ] <variable>

```

4.2.5.2 IS SelectFunction

The optional IS *SelectFunction* offers a set of functions to the end user. The IS *SelectFunction* extends the functionality of the current state by providing functions which are applied to the main object of the state (as referred to by the primary IS). The IS *SelectFunction* is associated with a set

¹ Development toolkits which support the creation of a MyUI AAIM should guarantee this association by automatically generating outgoing transitions for certain interaction situations defined in the states (the MyUI development toolkit is described in deliverable D3.3).

of application functions to modify the application data. Each application function can be triggered by a specific user interface element with a unique identifier (can be labels, images, voice commands or combinations of them). If the text label of the user interface element is used as identifier it is marked by being enclosed by double quotes.

For specification in a state of the AAIM chart, the formal notation of the IS *SelectFunction* is defined as follows:

$\langle IS\ SelectFunction \rangle ::= 'IS(SelectFunction),' \langle data\ provider \rangle$
 $\langle data\ provider \rangle ::= '([(\langle identifier \rangle ',') *] \langle identifier \rangle)'$

Whereas the order of the identifiers as given by the data provider determines the order in which the related user interface elements are displayed on the user interface.

Every application function as addressed by the IS *SelectFunction* is associated with exactly one outgoing transition from the linked state. The transition is triggered when the end user selects the respective identifier (i.e. user interface element). Every time when the transition is triggered, the related application function is executed.

A transition which belongs to an IS *SelectFunction* is labelled in the following notation format:

$\langle transition \rangle ::= 'SelectFunction:' \langle identifier \rangle '/' \langle application\ function \rangle$
 $\langle application\ function \rangle ::= \langle function\ name \rangle '([\langle function\ parameters \rangle])'$
 $\langle function\ parameters \rangle ::= [(\langle parameter \rangle ', ')*] \langle parameter \rangle$
 $\langle parameter \rangle ::= \langle variable \rangle / 'IS(' \langle primary\ IS\ name \rangle ', ' \langle data\ provider \rangle)'$
 $\langle data\ provider \rangle ::= \langle direct\ content\ input \rangle * / \langle data\ acquisition\ functions \rangle$
 $\langle data\ acquisition\ functions \rangle ::= [(' _ ' \langle function \rangle ', ')*] (' _ ' \langle function \rangle)$
 $\langle function \rangle ::= \langle function\ name \rangle '([\langle function\ parameters \rangle])'$
 $\langle function\ parameters \rangle ::= [(\langle parameter \rangle ', ')*] \langle parameter \rangle$

4.2.5.3 IS *SelectGoTo* and IS *MetaGoTo*

The optional IS *SelectGoTo* and the IS *MetaGoTo* offer navigation options to the user which lead to other states without changing the application data. Each target state that can be reached in this way results from the usage of a certain user interface element with a unique identifier (similar to the IS *SelectFunction* labels, images, voice commands or combinations of them are possible). Again, text labels used as identifiers are enclosed by double quotes.

The notation of the IS *SelectGoTo* (and the IS *MetaGoTo* respectively) in a state of the MyUI AAIM is formally defined as follows:

$\langle IS\ SelectGoTo \rangle ::= 'IS(SelectGoTo),' \langle data\ provider \rangle$
 $\langle data\ provider \rangle ::= '([(\langle identifier \rangle ',') *] \langle identifier \rangle)'$

Whereas again the display order of the related user interface elements follows the order of the provided identifiers.

Every identifier is associated with one outgoing transition from the related state. The state transition is triggered when the end user selects the respective identifier (i.e. user interface element). No additional function or activity is executed.

SelectGoTo transitions (and *MetaGoTo* transitions respectively) are labelled in the following format:

$\langle transition \rangle ::= 'SelectGoTo:' \langle identifier \rangle$

4.2.6 References to functions

As described in section 4.1 and in the definitions above, the AAIM provides references to functions which are defined in the application code (not in the AAIM). The referred functions can be

- *Data acquisition functions* to deliver the content for the current interaction situation, e.g. the items of a selection list in the IS selectItem or
- *Application functions* which are triggered when a specific transition from one application state to another is executed or when a state is entered. Application functions can set certain system variables, manipulate the application data base or perform transactions as, for example, send an email.

All application functions names and variables use lower camel case and start with a verb, e.g. createNewEmail(). In case of data acquisition functions a '_' is added at the beginning, e.g. _getEmailsFromServer.

4.3 Source code generation from the AAIM

The MyUI infrastructure for adaptive user interfaces automatically transforms an AAIM into an application skeleton with associated files and folders. An overview about the relations between the graphical view of the application in form of an example AAIM and the generated code is given within this section.

An exemplary MyUI AAIM is presented below. The following AAIM shows different states, transitions, interaction situations and data acquisition functions.

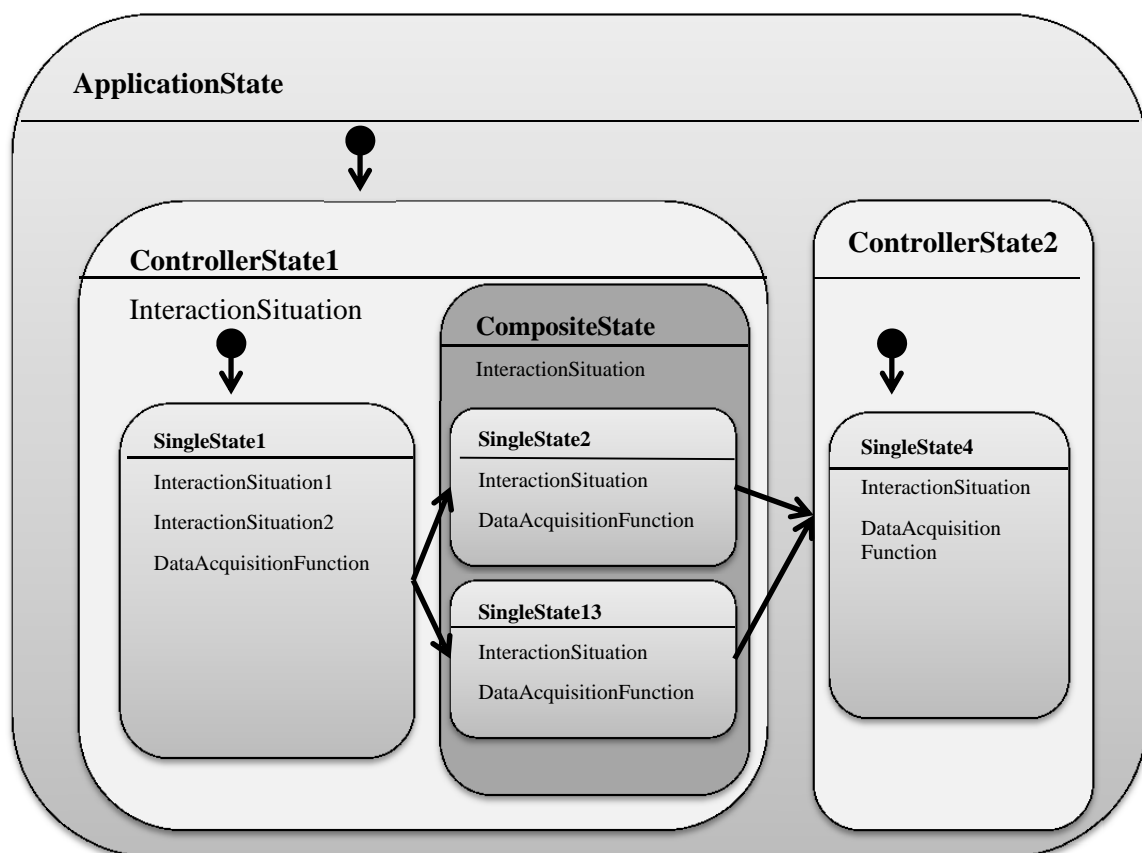


Figure 13 Basic concepts in the MyUI AAIM (example)

Every object of this AAIM has a certain representation in the source code. In the following all major components of the AAIM are introduced and their purpose in the Model View Controller design paradigm is explained:

- *Application State:*
The Application state is the main state of the application. All application specific variables which are affected by the underlying states are specified here. The entry point (default controller) of the application is also defined in the application state.
- *Composite State:*
As described above, view states can be contained in one composite state to enable a more compact representation of the AAIM. Interaction situations which are defined in a composite state are referenced by all the clustered view states.
- *Controller State:*
Controller States are also composite states which can contain single or clustered. Like the controller in MVC a controller state in the AAIM contains application functions as well as data acquisition functions and passes the result to a certain view. Data acquisition functions which are implemented in the controller are link with a certain data source which represents the model in MVC. As explained in section 3.6, the controller handles the user input, relates to the underlying data source and is responsible for notifying the associated View States which displays the current value of the model to the user. The AAIM can cover more than one controller state to cluster associated functions which relates to a certain model. The Controller state defines also a default view state which appears as the initial user interface when the controller state is entered.
- *Single State:*
Theses states are the user interface representation in the AAIM. Single States in the AAIM have the same behaviour as views in MVC. They contain one or more interaction situations and certain data sources which are presented to the user. Single states contain also references to data acquisition functions which are defined and executed in the overlying controller state.

4.4 Example: AAIM for the MyUI email application

The following Figure 14 depicts the high level navigation in the MyUI email application. All presented states are composite states. Figure 15 provides the detailed view of the composite “folders”.

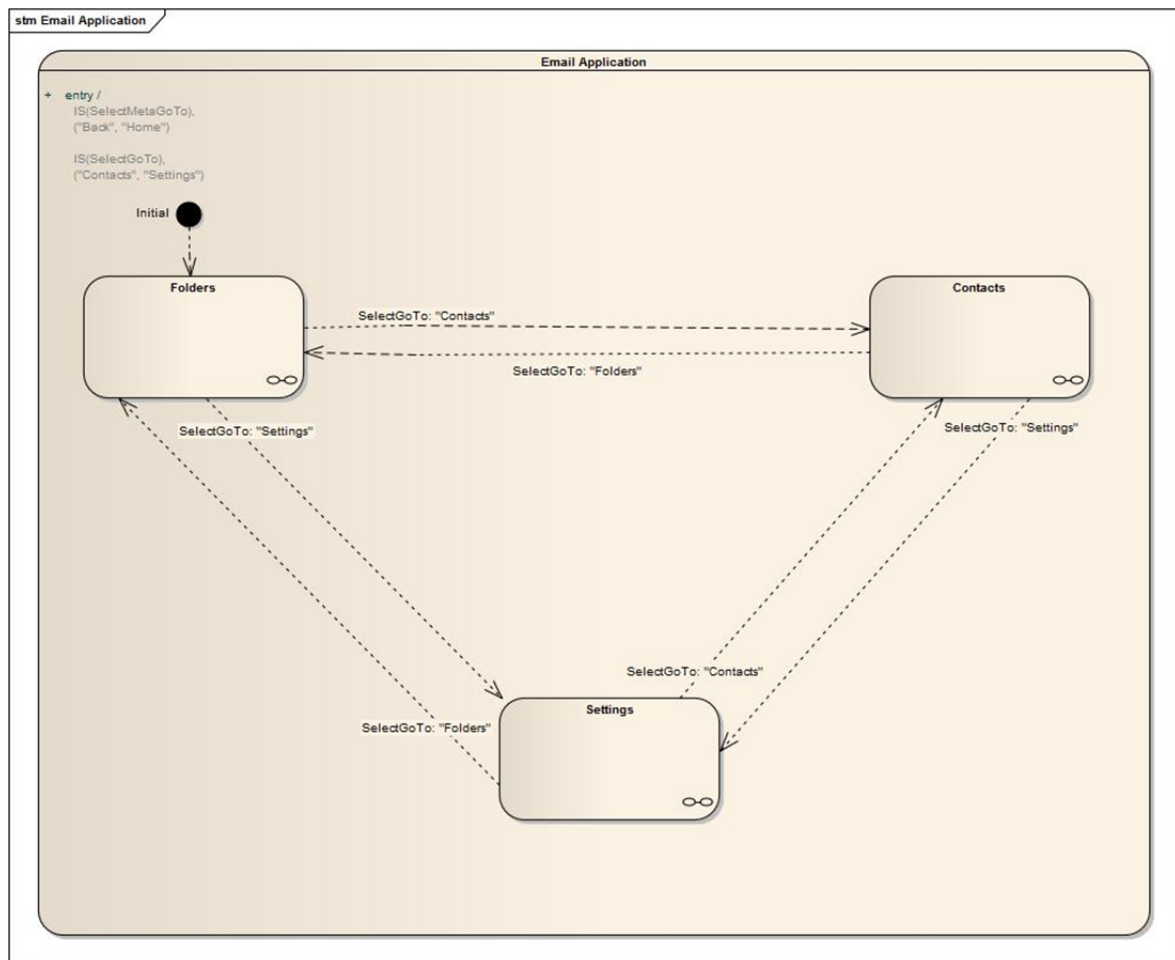


Figure 14 AAIM for the MyUI email application – part 1 (high level)

For a better understanding of Figure 15, the primary interaction situations (IS) used in the folder sub states are briefly described in the following (see the MyUI Patterns Repository at <http://myuipatterns.clevercherry.com> for more details):

- **IS SelectItemWithAttributes**
provides a list of items for single selection; each item is presented with respective meta-data (attributes).
- **IS ViewItemWithAttributes**
displays an information element (e.g. email, photo, song, news) together with related meta-data (e.g. sender, date, size).
- **IS EditForm**
provides a form with a number of entry fields and selection elements.
- **IS NavigateTree**
provides an interactive overview of elements which are structured in a hierarchical manner for single selection
- **IS ProvideFunctionAttributes**
retrieves additional information from the user which is required for the execution of a selected function, e.g. by a message box asking for the name of a folder to which an email should be moved to.

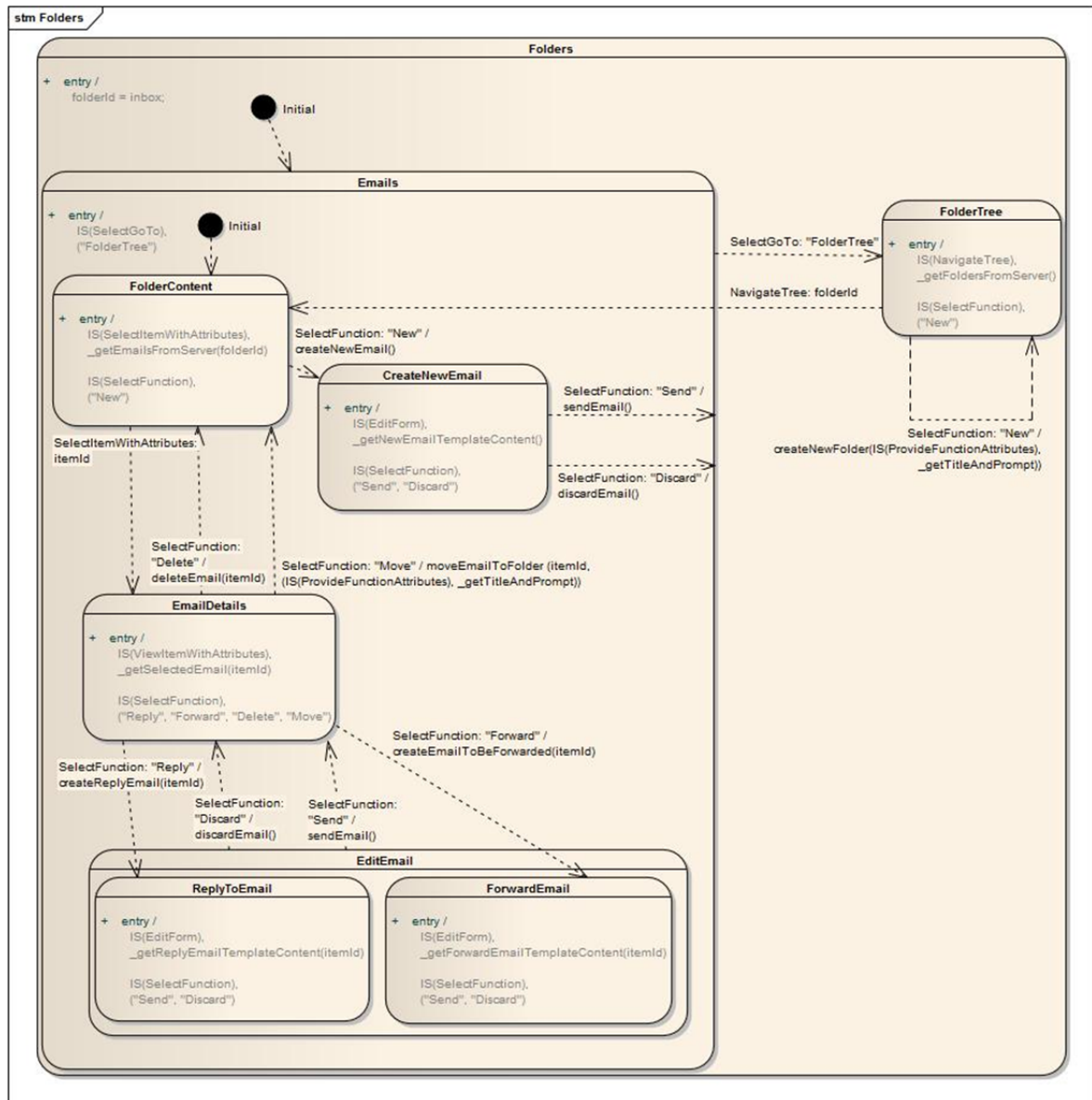


Figure 15 AAİM for the MyUI email application – part 2 (details, low level)

5. MyUI Design Patterns for Adaptive User Interfaces

5.1 Design Patterns as Building Blocks of a Modular User Interfaces

To cover the great heterogeneity of users, MyUI follows a modular approach to user interface development which relies on the composition of multimodal user interface design patterns. The design patterns provide proven solutions for specific interaction contexts, specific end user characteristics and contextual requirements which are explicitly addressed by the design. Each pattern is associated with software components that can be composed and modified in order to achieve accessible user interfaces. Thus, individual accessibility is achieved by combining design patterns that suit for a certain end user and context of use. User interface adaptation is done by switching from one instance of a design patterns cluster (e.g. all patterns for single selection from a list of options) to another design pattern of the same cluster which is hypothesized to fit best to the individual user's needs.

The conceptual back bone of the MyUI is the design patterns repository which includes proven design solutions for optimal accessibility and usability. The MyUI Design Patterns Repository is maintained as a media wiki in order to assure a high level of flexibility and extensibility (see <http://myuipatterns.clevercherry.com>). Each pattern is described in a defined structure as proposed by Borchers (2001) and related to other patterns of different types and levels of abstraction.

5.2 Types of Design Patterns in the MyUI Design Patterns Repository

The MyUI design patterns repository includes six categories of design patterns. Each pattern type fulfils distinct functions in the MyUI adaptation framework (see Figure 16):

1. *Device-specific patterns,*
2. *Individualization patterns,*
3. *Interaction patterns,*
4. *User interface elements,*
5. *Adaptation rendering patterns and*
6. *Adaptation dialogue patterns.*

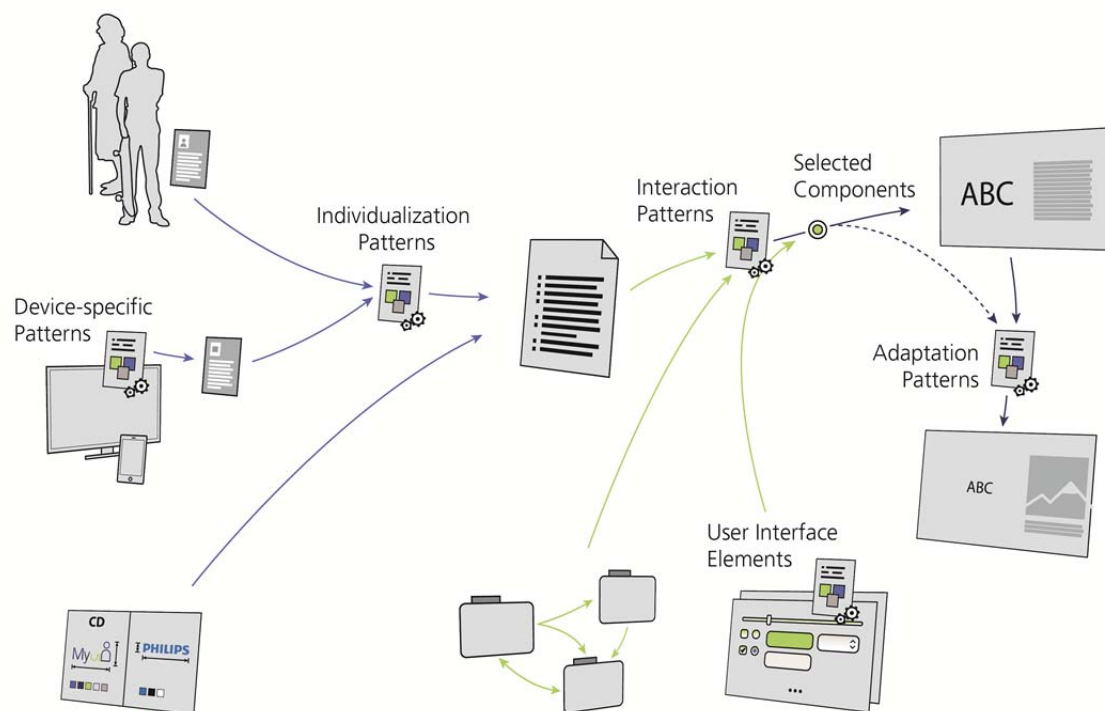


Figure 16 Design patterns in the MyUI adaptation framework

Device-specific patterns create a device profile on the basis of primitive device features as provided by the device. The variables of the device profile can be regarded as global variables which prepare the step of user interface parameterization by providing device-specific ranges of user interface settings from which individualization patterns select the most suitable (e.g. potential font sizes). Device-specific patterns define device-related user interface features which are stable throughout the interaction with a certain set of input and output devices. If the device setup is changed within one session, the device profile will be updated accordingly. As a result, user interface adaptations can be necessary also during use.

Individualization patterns define global user interface settings to fit individual user needs and specific context conditions, e.g. big font size for users with low vision or mainly iconographic display style for users with problems in language reception. As the available knowledge about user needs and context conditions can change during the interaction (user profile updates), individualization patterns play a role also in user interface adaptations during use.

These first two pattern types work on a general level of the user interface. Together, they are responsible for setting and adjusting global variables which apply to the entire user interface and not only to a specific interaction situation. Their contribution to the MyUI adaptation process finds expression in the User Interface Profile and is described in further detail in section 6 “User Interface Parameterization”. An additional input to the User Interface Profile comes from the Customization Profile which defines user interface settings to fit a specific corporate, project or brand identity, e.g. colours or font styles. The Customization Profile settings are directly used in the User Interface Profile in a 1:1-manner without transformations. Therefore, no specific patterns are needed for customization.

Interaction patterns provide suitable user interface components for a current interaction situation as specified in the AAIM, e.g. a list element for an interaction situation in which a user can select from a set of options. For each interaction situation, a bundle of several different interaction design patterns exists. All patterns of one bundle have the same purpose and suit the same interaction

situation. But they differ in appearance or input mechanisms in order to support different user needs and context conditions. The selection of the best suitable interaction pattern from a current bundle is done on the basis of specific variables of the User Interface Profile.

User interface elements are the basic building blocks for the currently selected interaction patterns. While interaction patterns can be regarded as components to support a given interaction situation, the user interface elements provide the generic primitives required to compose the interaction patterns, e.g. a “selection list” interaction pattern requires “option buttons” as user interface elements.

Interaction patterns and user interface elements are the components and elements from which a specific MyUI user interface is composed. The selection of the adequate user interface solutions is done by the interaction patterns. Input to this selection process includes the interaction situations of the AAIM and the User Interface Profile. Moreover, each interaction situation triggers a set of required user interface elements. The process of selecting the most suitable user interface components and elements is called “User Interface Preparation” and is further described in section 7. The rendering and updating of the complete user interface is described in section 8 (User Interface Generation and Adaptation).

Adaptation patterns cover the dynamics of the adaptation processes, i.e. they define the mechanisms of switching from one instance of a user interface to another. Whereas *adaptation rendering patterns* are responsible for the low-level definition of rendering the transitions of single display and control elements, *adaptation dialogue patterns* specify the dialogue between the adaptive user interface and the end user in the course of an adaptation. The range of potential adaptation dialogue patterns goes from requesting explicit user confirmations before adapting the user interface to automatic and sudden changes without directed user action. Besides system-initiated adaptations, the adaptation dialogue patterns cover also instances where the end user initiates a change in the user interface by explicitly modifying the user profile or the user interface profile. The design and selection of suitable adaptation patterns is crucial for the usability, understandability and acceptability of adaptive user interfaces. Changes in different areas of the user interface will require different levels of user awareness and user initiative in order to be effective.

5.3 MyUI Design Pattern Language

According to Borchers’ understanding, patterns are not isolated but refer to other patterns. In a hierarchical pattern language, larger patterns refer to smaller-scale patterns for the solution they describe. And patterns can only be used in a certain type of context which is the result of applying larger-scale patterns (cf. Borchers, 2001).

In the MyUI design pattern language, references between patterns play an important role. For a systematic use of references a classification of relations between patterns has been defined. The following relations are used in the MyUI pattern language:

5.3.1 Relation *substitutes*

Definition

Pattern A *substitutes* pattern B, if both patterns A and B serve the same purpose in the MyUI framework and if both patterns can never be active at the same time. Purposes in the MyUI framework include setting a variable of the user interface profile, applying to an interaction situation of the AAIM and providing a user interface adaptation mechanism. The relation *substitutes* is symmetric, i.e. A *substitutes* B if and only if B *substitutes* A.

Purpose

The *substitutes* relation is used in order to group patterns which serve the same general purpose but support different user needs or context conditions, e.g. apply to different conditions in the profiles

used in MyUI (e.g. user profile, user interface profile). The *substitutes* relation helps to structure the MyUI pattern repository by creating bundles of patterns with the same purpose. The patterns of a bundle can be called variants of the bundle. Typically, each bundle has a default pattern which applies to the default settings of the relevant profile.

Application

The *substitutes* relation can be applied to the following pattern types:

- Individualization pattern *substitutes* individualization patterns (of the same bundle)
- Device-specific pattern *substitutes* device-specific patterns (of the same bundle)
- Interaction pattern *substitutes* interaction patterns (of the same bundle)
- Adaptation rendering pattern *substitutes* adaptation rendering patterns (of the same bundle)
- Adaptation dialogue pattern *substitutes* adaptation dialogue patterns (of the same bundle)

5.3.2 Relation *requires*

Definition

Pattern A *requires* pattern B,
if pattern B describes parts of the higher-level solution addressed by pattern A, i.e. the implementation of pattern A requires pattern B for a detailed specification of parts of the overall solution.

Set A of patterns *requires* pattern B,
if pattern B describes parts of the higher-level solution addressed by the pattern of set A.

In practice, the *requires* relation results in automatically triggering the required pattern B when pattern A or set A of patterns is selected.

Purpose

The *requires* relation is used to structure the design patterns repository in a vertical way. It links higher-level patterns which describe entire user interface components to lower-level patterns detailing the provided solution by describing the used generic user interface primitives.

Application

The *requires* relation can be applied to the following pattern types:

- Interaction pattern *requires* user interface element
- User interface element *requires* user interface element
- Ordered triple of one adaptation dialogue pattern and two interaction patterns of the same bundle *requires* adaptation rendering pattern.
- Ordered pair of one adaptation dialogue pattern and a user interface element *requires* adaptation rendering pattern.

5.3.3 Relation *is required by*

Definition

Pattern A *is required by* pattern B,
if pattern A describes parts of the higher-level solution addressed by pattern B.

Pattern A *is required by* set B of patterns,
if pattern A describes parts of the higher-level solution addressed by the patterns of set B, whereas the patterns of set B can be of different pattern types.

is required by is the inverse relation of *requires* (cf. section 5.3.2).

Purpose

The *is required by* relation is used for a vertical structuring of the design patterns repository. It links lower-level patterns which describe generic user interface primitives to the higher-level patterns making use of them in describing composed solutions.

Application

The *is required by* relation can be applied to the following pattern types:

- User interface element *is required by* interaction pattern
- User interface element *is required by* user interface element
- Adaptation rendering pattern *is required by* an ordered triple of one adaptation dialogue pattern and two interaction patterns of the same bundle
- Adaptation rendering pattern *is required by* an ordered pair of one adaptation dialogue pattern and a user interface element.

5.3.4 Relation *sets <variables> as required by*

Definition

Pattern bundle A *sets <variables> as required by* pattern bundle B, if the patterns of bundle A set one or more global MyUI variables which specify (parts of) the solutions as provided by the patterns of bundle B. As a consequence, the variables of the relation (as set by A) are referred to in the solution field (then-statement) of the pattern descriptions in bundle B.

Purpose

The *sets <variables> as required by* relation is used to link pattern bundles A and B of different types within the user interface parameterization process. It supports the stepwise adaptation of a specific user interface profile variable which first applies to a certain device and a project or corporate identity, and then to an individual user.

Application

The *sets <variables> as required by* relation can be applied to the following pattern types:

- Bundle of device-specific pattern *sets <variables> as required by* bundle of individualization pattern

5.3.5 Relation *requires <variables> as set by*

Definition

Pattern bundle A *requires <variables> as set by* pattern bundle B, if the patterns of bundle A require for their solutions one or more global MyUI variables which are set by the patterns of bundle B (as parts of). As a consequence, the variables of the relation (as set by B) are referred to in the solution field (then-statement) of the pattern description of bundle A.

requires <variable(s)> as set by is the inverse relation of *sets <variable(s)> as required by* (cf. section 5.3.4).

Purpose

The *requires <variables> as set by* relation is used to link pattern bundle A and pattern bundle B of different types within the user interface parameterization process. It supports the stepwise adaptation of a specific user interface profile variable which first applies to a certain device and a project or corporate identity, and then to an individual user.

Application

The *requires <variables> as set by* relation can be applied to the following pattern types:

- Bundle of individualization patterns *requires* <variables> as set by bundle of device-specific pattern

5.3.6 Relation *sets* <variable(s)> as used by

Definition

Pattern bundle A *sets* <variable(s)> as used by pattern bundle B, if the patterns of A set one or more global MyUI variables which are used to select the most suitable pattern from bundle B. As a consequence, the variables of the relation (as set by A) are referred to in the context field (if-statement) of the pattern descriptions of bundle B.

Purpose

The *sets* <variables> as used by relation is used to link pattern bundles A and B of different types. Pattern bundle A engaged in creating or updating one of the used profiles (i.e. device profile or user interface profile) is associated with bundle B by one or more global variables which are crucial for selecting the most suitable variant of pattern bundle B. Thus, the relation *sets* <variables> as used by supports the horizontal structure of the MyUI pattern repository by reflecting the strong link between the sequential steps of device profiling, user interface parameterization and user interface preparation in the MyUi adaptation process.

Application

The *sets* <variables> as used by relation can be applied to the following pattern types:

- Bundle of device-specific patterns *sets* <variables> as used by bundle of device-specific patterns
- Bundle of device-specific patterns *sets* <variables> as used by bundle of individualization patterns
- Bundle of individualization patterns *sets* <variables> as used by bundle of interaction patterns

5.3.7 Relation *uses* <variable(s)> as set by

Definition

Pattern bundle A *uses* <variable(s)> as set by pattern bundle B, if the pattern bundle A uses one or more global MyUI variables as set by pattern bundle B in order to select the most suitable pattern from A. As a consequence, the variables are referred to in the context field (if-statement) of the pattern descriptions in bundle A and in the solution field (then-statement) in bundle B.

uses <variable(s)> as set by is the inverse relation of *sets* <variable(s)> as used by (cf. section 5.3.5).

Purpose

The *uses* <variable(s)> as set by relation is used to link pattern bundles A and B. A bundle A of patterns is associated with a bundle of profiling/parameterization patterns B which set global variables crucial for selecting the most suitable variant of pattern bundle A. Similarly to its inverse relation, this patterns supports the horizontal structure of the MyUI pattern repository by reflecting the strong link between the sequential steps of device profiling, user interface parameterization and user interface preparation in the MyUi adaptation process.

Application

The *uses* <variables> as set by relation can be applied to the following pattern types:

- Bundle of device-specific patterns *uses* <variables> as set by bundle of device-specific patterns

- Bundle of individualization patterns *uses <variables> as set by* bundle of device-specific patterns
- Bundle of interaction patterns *uses <variables> as set by* bundle of individualization patterns

5.4 MyUI Design Patterns Repository

The MyUI patterns repository includes the MyUI design patterns in a human-readable description format and re-usable software components. The MyUI design patterns describe solutions for specific design problems in interaction design for end users with special needs. Each type of design patterns addresses a specific level of user interface design as described in the sections above (cf. section 5.2). Each design pattern describes a specific part of a solution for a certain interaction situation and certain end user needs. Every instance of a MyUI user interface is the result of the composition of a number of single design patterns – or clearly put: reusable software components linked to the design patterns. Each of the design patterns is associated with a source code representation of the described design solution to put the recommended guideline into action in the MyUI adaptive user interface. With both, a machine-readable and a human-readable part, the MyUI design patterns repository bridges the gap between guidelines for accessible design and generative user interfaces.

Extensibility is an important aspect of the design patterns repository. The current body of design patterns in MyUI reflects the interaction requirements of the MyUI demonstrator applications, i.e. email application and instant messaging. For future applications, additional patterns will be needed. These can be simply added to the pattern repository without requiring a new version of a monolithic patterns document. Also new knowledge about and experiences with the current patterns can lead to modifications and refinements of the current body of design guidelines. Therefore, the design patterns repository is not a static part of this deliverable. This document describes the conceptual framework of user interface adaptation in MyUI. Furthermore, it explains the pattern-based approach of modular user interface generation and adaptation. The different pattern types and their roles in the adaptation process are introduced and described in detail.

However, the content of the single patterns is not replicated in this document. Please visit the MyUI Pattern Browser at <http://myuipatterns.clevercherry.com> for an up-to-date version of the pattern repository which will be evolved and refined also after the submission of this document and even after the official end of this project (see Figure 17). The MyUI Pattern Browser is developed in work package 3 as one of the tools for MyUI developers. It provides a structure and guidance for using the human-readable part of the design patterns. References to the re-usable software components of each of the patterns are provided. Moreover, the pattern browser includes general information on the MyUI adaptation process and the technical artefacts such as the user profile, the device profile, and the user interface profile. It is conceptualized as an open wiki for developers who can help to improve the body of knowledge, add new patterns, etc.

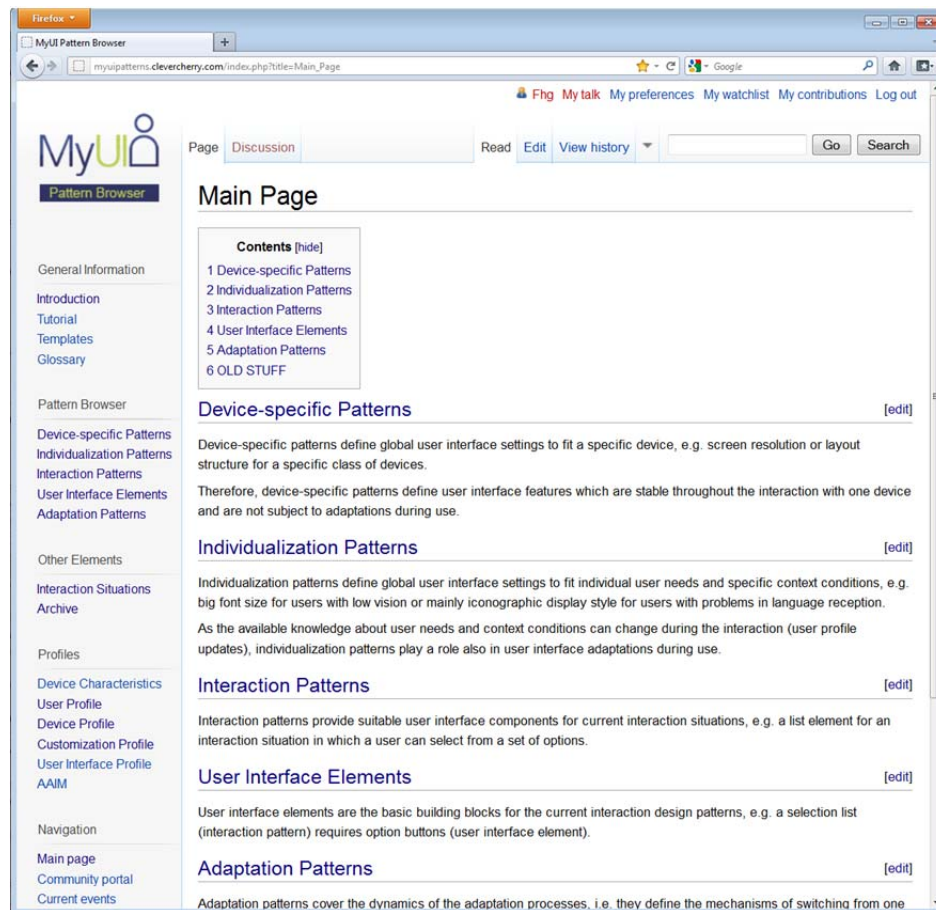


Figure 17 The MyUI Pattern Browser is the developer's access to the MyUI Pattern Repository

6. User Interface Parameterization

User interface parameterization is the first step in the MyUI user interface generation and adaptation process. The result (output) of this first step is the MyUI User Interface Profile which defines general characteristics of the user interface. The settings of the user interface profile are valid throughout the entire user interface and in all interaction situations of an application.

UI parameterization processes information from three different input sources:

1. Information about the currently available and used device set up:
the Device Manager provides information about the current devices via the *Device Profile*.
2. Information about the user and the current environment of use:
this information is drawn from the *User Profile* as described in D1.2. Mechanisms of creating and updating a user profile are part of the MyUI Context Management Infrastructure developed in WP 1 (cf D1.2).
3. Customization settings specific to the product, project, brand or company:
the developer of a MyUI application can create a *Customization Profile* which results in a product-, project-, brand- or corporate-specific appearance of the adaptive user interface.

User Interface Parameterization can be regarded as a transformation of these data sources and profiles into the user interface profile. The following pattern types are involved in this transformation process:

- *Device-specific patterns*
input: available knowledge about the current input and output devices,
output: device profile
- *Individualization patterns*
input: device profile and user profile,
output: user interface profile (except for the customization-related variables)

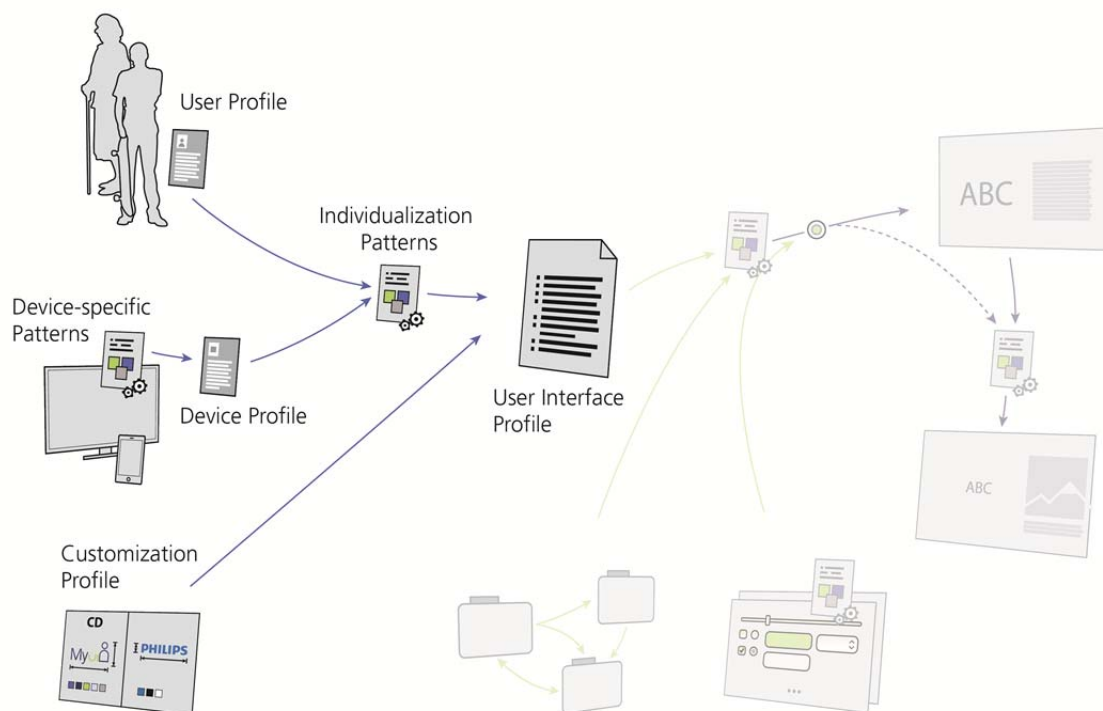


Figure 18 Profiles and patterns in the MyUI user interface parameterization

Figure 18 above presents an overview of the user interface parameterization process. In the following sections the involved profiles, mechanisms and patterns are described in more detail.

6.1 Device Manager

The device manager has two major tasks in the user interface parameterization process:

1. The first step of the parameterization process is to register the available interaction devices when the interaction between an end user and a MyUI application starts. The device manager includes a device repository with all potential MyUI devices and their specifications. Firstly, the Device Manager gathers all available information directly from the device such as screen resolution, etc. Then, it draws additional detailed specifications about the capabilities of the currently registered input/output devices from the device repository.
2. In a second step, the device manager transforms the collected device specifications into a device profile. Device-specific patterns categorize the available devices and set variables of the device profile accordingly. A detailed description of the Device Profile and its variables as well as their effect on the user interface parameterization process is described in the next section. Similar to the context management process (cf. WP 1) the device manager is working in a continuous process. For example, if the end user changes the device orientation from landscape to portrait, then device-specific patterns change specific variables of the device profile which, as a consequence, can affect the user interface profile. The following Figure 19 shows how the device manager interacts and transforms device specifications into device profile variables.

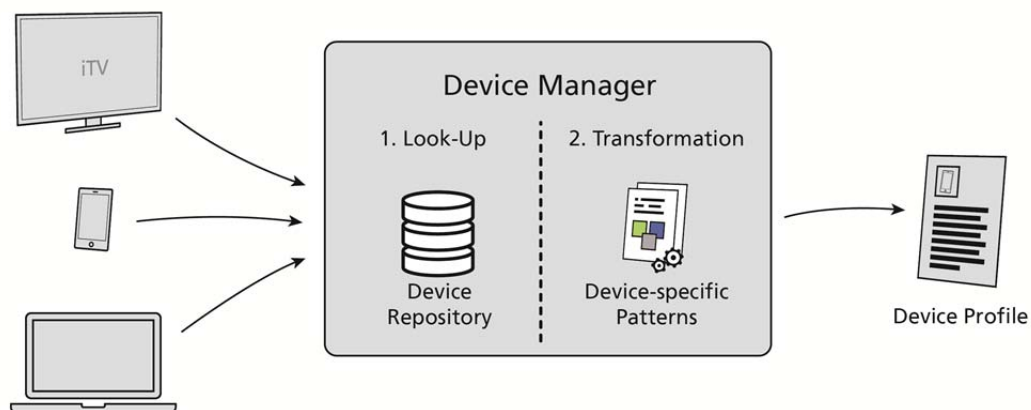


Figure 19 The device manager creates and updates the device profile via device-specific patterns

6.2 Device Profile

The device profile is created and updated by the device manager via device-specific patterns. It contains the following variables which specify the used devices:

- **Category**
The category specifies the class of the device on which the MyUI application is running like for example smart phone, iTV, computer, tablet etc. This variable provides certain basic information about the context of use such as the viewing distance to the screen.

- **Font Variables**
Depending on the category (smart phone, iTV computer etc.) different font sizes are set as constants for the user interface (e.g. titleFontSize, bodyTextFontSize and complementaryTextFontSize).
- **Layout Variables**
According to the output characteristics of the device layout parameters are set as variables in the device profile. These include variables to specify the size and number of the layout grid cells (cellSize and cellCount), the orientation of the screen (landscape or portrait) and the coordinates of different screen areas (TitleBar, ContentArea, FunctionArea etc.).
- **Input devices**
The available input devices are also specified in the Device Profile. On the basis of these variables the user interface offers different interaction methods to the user. For example if the variables microphoneAvailable and keyboardAvailable are set to true, then the system can offer user interaction via speech, numeric and arrow navigation.
- **Output devices**
The last group of variables specify the available output devices. The current Device Profile consists of two variables which define if a visual display and speakers are available.

The following Table 1 shows an example for a device profile of an iTV.

Table 1 Example for a MyUI device profile for an iTV setup

Device variable	Value	Description
<i>category</i>	iTV	Category of the device on which the application is running
Font variables		
<i>titleFontSize</i>	{small: 36 pt; medium: 55 pt; large: 83 pt; xlarge: 110 pt}	Font size of title elements
<i>bodyTextFontSize</i>	{small: 24 pt; medium: 34 pt; large: 51 pt; xlarge: 68 pt}	Font size of all body text elements
<i>complementaryTextFontSize</i>	{small: 18 pt; medium: 21 pt; large: 32 pt; xlarge: 42 pt}	Font size of all complementary text elements, e.g. additional descriptions or comments of minor importance for the user
Layout variables		
<i>cellSize</i>	5 px	The variable grid describes how many pixels are needed to build one cell of a grid.
<i>cellCount</i>	128x72	The variable cellCount describes how many cells are needed to fill the display.
<i>titleBarArea</i>	{from: 0x0; to: 111x12; hide: false; visible: true}	The Coordinates and layout properties of the title bar
<i>contentArea</i>	{from: 13x15; to: 84x59; hide: false; visible: true}	The Coordinates and layout properties of the content area
<i>contentControlArea</i>	{from: 13:60; to: 84x72; hide: false; visible: true}	The Coordinates and layout properties of the content control area
<i>functionsArea</i>	{from: 90:15; to: 122:66; hide: false; visible: true}	The Coordinates and layout properties of the functions area
<i>adaptationArea</i>	{from: 90:66; to: 122x72; hide: false; visible: true}	The Coordinates and layout properties of the adaption area
Input Devices		
<i>microphoneAvailable</i>	{available: true; type: external}	Whether there is a microphone available
<i>touchAvailable</i>	false	Whether there is a touch input device available

<i>pointingDeviceAvailable</i>	true	Whether there is a pointing device available
<i>keyboardAvailable</i>	{available: true; type: QWERTY}	Whether there is a keyboard available
<i>remoteControlAvailable</i>	true	Whether there is a remote control available
<i>cameraAvailable</i>	true	Whether there is a camera available
Output Devices		
<i>displayAvailable</i>	true	whether there is a display available
<i>speakerAvailable</i>	true	whether there is a speaker available

6.3 Device-Specific Patterns

Device-specific patterns are responsible for processing the transformations from primitive device features to variables of the device profile in the device manager. They receive their input directly from the device which communicates general meta-data and from the device repository (cf. section 6.1). Device-specific patterns are triggered at the beginning of a new session and when the device setup is changed during a session, e.g. by adding or removing an input/output device. As an output, device-specific patterns set and update the variables of the device profile. The device profile settings are valid for all users in all situations. The device profile variables are used as input variables for the individualization patterns (see section 6.6 below).

Device-specific patterns can have the following relations with other patterns (cf. section 5.3):

- *Substitutes*
grouping two or more device-specific patterns to a pattern bundle
- *sets <variables> as required by* (and the respective inverse relation)
supporting the stepwise adaptation of a specific user interface profile variable which first applies to a certain device and then to an individual user, e.g. the device-specific pattern »Font Size – iTV« sets the variable *dp.bodyTextFontSize* which is referred to in the solution described by the individualization patterns »Font Size«. This relation is defined on the level of device-specific pattern bundles.
- *sets <variables> as used by* (and the respective inverse relation)
reflecting the sequential steps of device profiling, user interface parameterization and user interface preparation in the MyUi adaptation process, e.g. the device-specific patterns »Categorize Device« set the variable *dp.deviceCategory* which is input to the device-specific pattern bundle »Layout Grid and Structure« to determine a suitable layout grid and layout structure for the current device. This relation is defined on the level of device-specific pattern bundles.

In order to avoid redundant specifications in all device-specific patterns of a bundle, the MyUI design patterns repository includes bundle descriptions for device-specific patterns in the following format:

Table 2 Template for device-specific pattern bundles

<ID>	<name of the device-specific pattern bundle>
Problem	<describe a problem which has to be resolved for the entire application or for all applications on a given device.>
sets <variable(s)> as required by <pattern bundle>	sets <variable(s)> as required by <pattern bundle>
sets <variable(s)> as	sets

used by <pattern bundle>	<variable(s)> as used by <pattern bundle>
Patterns	<indicate the patterns of the bundle>

The following Figure 20 shows an example of a device-specific pattern bundle description in the MyUI patterns repository.

Device Font Sizes	
Pattern Bundle	Device Font Sizes
Problem	Within this pattern bundle, there are patterns for each available device. Within each pattern, all relevant font sizes for this device are defined
State (temporarily only)	1
sets <variable(s)> as required by <pattern bundle>	sets <i>titleFontSizeSmall</i> <i>bodyTextFontSizeSmall</i> <i>complementaryTextFontSizeSmall</i> <i>titleFontSizeMedium</i> <i>bodyTextFontSizeMedium</i> <i>complementaryTextFontSizeMedium</i> <i>titleFontSizeLarge</i> <i>bodyTextFontSizeLarge</i> <i>complementaryTextFontSizeLarge</i> <i>titleFontSizeXLarge</i> <i>bodyTextFontSizeXLarge</i> <i>complementaryTextFontSizeXLarge</i> as required by Font Size
sets <variable(s)> as used by	
Patterns	Font Size iTV

Figure 20 Device-specific pattern bundle description (example)

The template for device-specific patterns includes the following fields:

Table 3 Template for device-specific patterns

<ID>	<name of the device-specific pattern>
Problem	<describe a problem which has to be resolved for the entire application or for all applications on a given device.>
Pattern Bundle	< pattern bundle>
Context	IF <check device specification values>
Solution	THEN Set <device profile variable(s)> = <value(s)>
Code reference	<if applicable, provide a reference to the related code to implement the solution>
Diagram	<if applicable, illustrate the design solution in a schematic and concise manner>
Rationale (references)	<explain the principles or rationale behind the pattern and provide

	references to literature, standards, etc.>
Substitutes	<device-specific pattern(s)>
uses <variable(s)> as set by <pattern bundle>	uses <variable(s)> as set by <pattern bundle>
requires <variable(s)> as set by <pattern bundle>	requires <variable(s)> as set by <pattern bundle>

The MyUI design patterns repository currently includes the following device-specific pattern bundles and patterns (cf. http://myuipatterns.clevercherry.com/index.php?title=Device-specific_Patterns):

- Categorize Device
 - Categorize iTV
- Device Font Sizes
 - Font Size iTV
- Layout Grid and Structure
 - Layout Structure – iTV
- Available Microphone
 - Available Microphone – true
 - Available Microphone – false
- Available Touch
 - Available Touch - true
 - Available Touch - false
- Available Pointing Device
 - Available Pointing Device - true
 - Available Pointing Device - false
- Available Keyboard
 - Available Keyboard - true
 - Available Keyboard – false
- Available Remote Control
 - Available Remote Control - true
 - Available Remote Control - false
- Available Camera
 - Available Camera - true
 - Available Camera - false
- Available Display
 - Available Display – true
 - Available Display - false
- Available Speaker
 - Available Speaker - true
 - Available Speaker – false

6.4 User Profile

The user profile stores information about the user and relevant environmental aspects such as ambient noise or ambient lighting conditions. The mechanisms for creating and refining the user profile are described in D1.2. The MyUI user profile includes the following variables (see D1.2 for a detailed description of the MyUI user profile):

Table 4 MyUI User Profile variables

Name of variable	Description	Value Space
Perceptual		
<i>Visual Acuity</i>	Ability to perceive what is displayed on the screen	[0,4]
<i>Field of Vision</i>	Describes how limited the field of vision of the given user is.	[0,4]
<i>Ambient Light</i>	The amount of ambient light at the users place.	[0,4]
<i>Ambient Noise</i>	The amount of ambient noise at the users place.	[0,4]
<i>Hearing</i>	Describes how limited the user's ability to hear sounds is.	[0,4]
Cognitive		
<i>Language Reception</i>	Ability to understand written or spoken language	[0,4]
<i>Language Production</i>	Ability to speak and write language	[0,4]
<i>Understanding Abstract Signs</i>	Ability to understand abstract signs and pictograms	[0,4]
<i>Attention</i>	Ability to handle multiple things at the same time, resp. focusing on something.	[0,4]
<i>Processing Speed</i>	Ability to process information fast.	[0,4]
<i>Working Memory</i>	Ability to remember an exact sequence of steps in a process and the ability to orientate in this process.	[0,4]
<i>Long Term Memory</i>	Ability to learn and remember information for a long time.	[0,4]
<i>ICT Literacy</i>	Ability to use modern information technology.	[0,4]
<i>Hand-Eye Coordination</i>	Ability to coordinate the movement of the hands with things seen.	[0,4]
Motor		
<i>Speech Articulation</i>	Ability to speak	[0,4]
<i>Finger Precision</i>	Ability to move the fingers precisely.	[0,4]
<i>Hand Precision</i>	Ability to move the hand precisely.	[0,4]
<i>Arm Precision</i>	Ability to move the arms precisely.	[0,4]
<i>Contact Grip</i>	Ability to control things by touching them.	[0,4]
<i>Pinch Grip</i>	Ability to press single buttons.	[0,4]
<i>Clench Grip</i>	Ability to hold object.	[0,4]
General		
<i>First Name</i>	The first name of the user.	
<i>Last Name</i>	The last name of the user.	
<i>Email Address</i>	The email address of the user.	
<i>Preferred Language</i>	The language the user prefers to use.	{English, German, Spanish}
<i>Successful Interactions</i>	The number of successful interactions with the system.	
<i>State transitions</i>	The number of state transitions the user carried out.	

<i>MyUI Experience</i>	The experience with the MyUI system.	[0, 4]
<i>PreferenceTonalOutput</i>	Selects whether the user prefers output enhanced with sounds.	{True; False}
<i>PreferenceSpeechOutput</i>	Selects whether the user prefers speech-output in addition to text.	{True; False}

6.5 User Interface Profile

The user interface profile includes all global settings of the user interface for a specific user at a certain time. These global settings encompass user interface variables such as font size or available, suitable and preferred interaction mechanisms (e.g. voice control, touch, gesture, etc.) which determine the overall appearance of the individualized user interface across all interaction situations. The creation and maintenance of a central user interface profile with global variables ensures a high consistency and an optimal adaptation of the user interface throughout all interaction situations. Adaptations that result from newly available information about a specific user need always do not result in modifying single user interface widgets but influence the appearance of the entire user interface.

The user interface profile includes the variables as listed below. Further variables will be added when additional input/output devices such as pointing devices or touch controls are supported. For each variable, the range of valid values is provided. Default values are highlighted in bold font.

Table 5 MyUI User interface profile variables

Global UI variable	Range	Description
<i>titleFontSize</i>	{ 36 pt ; 55 pt; 83 pt; 110 pt}	Font size of title elements
<i>bodyTextFontSize</i>	{ 24 pt ; 34 pt; 51 pt; 68 pt}	Font size of all body text elements
<i>complementaryTextFontSize</i>	{ 18 pt ; 21 pt; 32 pt; 42 pt}	Font size of all complementary text elements, e.g. additional descriptions or comments of minor importance for the user
<i>maxElementsPerScreen</i>	{ infinite ; 6;4; 3}	Restricts the number of interactive elements displayed in the content and functions area of the screen
<i>eyeCandy</i>	{ on ; off}	Defines if decorative elements are hidden or displayed
<i>cursorNavigation</i>	{ on ; off}	Defines if navigation with cursor keys is enabled or disabled
<i>numericNavigation</i>	{on; off }	Defines if navigation with numeric keys is enabled or disabled
<i>displayMode</i>	{text only; mainly text ; text and graphics; mainly graphics; graphics only}	Defines the proportion of text and graphics displays, refers to the two different codes of visual information presentation: verbal vs. spatial
<i>audioOutputVolume</i>	{ low ; medium; high; silent}	Defines the volume level of audio output and if audio output is enabled or disabled
<i>speechAudioOutput</i>	{on; off }	Defines if speech audio output is enabled or disabled
<i>speechAudioSpeed</i>	{ normal ; slow}	Defines the speed of speech audio output
<i>tonalAudioOutput</i>	{ on ; off}	Defines if tonal audio output is enabled or disabled
<i>voiceInput</i>	{ off ; on}	Defines if voice input is enabled or disabled
<i>skin</i>	{ MyUI ; ...}	Defines the Customization Profile to be selected. This variable is set by the developer in the development toolkit (project preferences).

6.6 Individualization Patterns

Individualization Patterns are the core piece of the automatic user interface individualization. They create and update the user interface profile. They are closely related to specific user characteristics as stored in the user profile and relevant device characteristics as stored in the device profile. They process the current user profile and device profile and “translate” the user, environment and device characteristics into user interface features, i.e. global settings.

With a blank user profile (i.e. no information available about the user), the default variants of all individualization patterns will be activated which is associated with the default user interface profile. Each time when changes occur to the user profile, all rules of the individualization patterns will be recalculated and the best fitting individualization patterns will be activated. This will result in changes to the global settings as defined in the user interface profile. Thus, user interface parameterization is a permanently on-going process during the interaction. Changes in the user profile serve as triggers for a parameterization updates.

Individualization patterns can have the following relations with other patterns (cf. section 6.3):

- *Substitutes*
grouping two or more individualization patterns to a pattern bundle
- *requires <variables> as set by*
establishing the connection to variables set by device-specific patterns which are referred to in the solution statement (then-statement) of a bundle of individualization patterns, e.g. the individualization patterns of the bundle »Font Size« refer to different variables of the vector *dp.bodyTextFontSize* (set by device-specific patterns) in order select the best fitting font size for an individual user.
- *sets <variables> as used by*
connecting individualization patterns with interaction patterns by variables of the user interface profile being the output of the one and the input of the other pattern, e.g. user interface profile variables such as *numericNavigation* determine which variant of the interaction pattern bundle »Main Menu« to select.
- *uses <variable(s)> as set by*
reflecting the sequential steps of device profiling and individualization and their dependencies in the user interface parameterization process, e.g. the user interface profile variable *numericNavigation* can only be set to on if a numeric key pad is available, i.e. if the device profile variable *remoteControlAvailable* (or *keyboardAvailable*) is true.

The MyUI design patterns repository includes bundle descriptions for individualization patterns in the following format:

Table 6 Template for individualization pattern bundles

<ID>	<name of the individualization pattern bundle>
Problem	< describe a general interaction problem related to a value range(s) of certain aspects/variables of the user profile and the device profile>
sets <variable(s)> as used by <pattern bundle>	sets <variable(s)> as used by <pattern bundle>
Patterns	<indicate the patterns of the bundle>

Individualization patterns are described in the following format:

Table 7 Template for individualization patterns

<ID>	<name of the individualization pattern>
Problem	< describe a general interaction problem related to a value range(s) of certain aspects/variables of the user profile and the device profile>
Pattern Bundle	<pattern bundle>
Context	IF <check values of user profile variable(s) and/or device profile variable(s)>
Solution	THEN Set <user interface profile variable(s)> = <value(s)>
Code reference	<if applicable, provide a reference to the related code to implement the solution>
Diagram	<if applicable, illustrate the design solution in a schematic and concise manner>
Rationale (references)	<explain the principles or rationale behind the pattern and provide references to literature, standards, etc.>
Substitutes	<individualization pattern(s)>
uses <variable(s)> as set by <pattern bundle>	uses <variable(s)> as set by <pattern bundle>
requires <variable(s)> as set by <pattern bundle>	requires <variable(s)> as set by <pattern bundle>

Figure 21 presents an example of an individualization pattern which determines the ratio of textual and graphical information on the screen in accordance with display the font size in accordance with current setting in the user profile variables *understandingAbstractSigns* and *languageReception*.

Display Mode – text and graphics

Individualization Pattern	Display Mode – text and graphics
Problem	Users with mild language reception impairments will benefit from a balanced display mode with text and graphics.
Pattern Bundle	Display Mode
Ranking	0
Context	If $(understandingAbstractSigns < 2$ AND $1 \leq languageReception < 2)$ OR $(understandingAbstractSigns \geq 2$ AND $languageReception \geq 2)$
Solution	Then set <i>displayMode</i> : text and graphics
Code reference	
Diagram	
Rationale (references)	See MyUI Document D2.1 (p. 57) URC11 citing ER-FG-01 (2010) National Institute on Aging (NIA) National Library of Medicine (2005) Making Your Website Senior FriendlyL A Checklist. http://www.nia.nih.gov/HealthInformation/Publications/website.html Kurniawan, S. and Zaphiris, P. (2005). Research-Derived Web Design Guidelines for Older People. Proceedings of 7th international ACM SIGACCESS Conference on Computers and Accessibility 20-05 (ASSETS'05), pp 129-135: http://portal.acm.org/citation.cfm?id=1090810
Substitutes:	Display Mode – mainly text Display Mode – mainly graphics Display Mode – graphics only Display Mode – text only
uses <variable(s)> as set by <pattern bundle>	
requires <variable(s)> as set by <pattern bundle>	

Figure 21 Individualization pattern description (example)

The MyUI design patterns repository currently includes the following individualization pattern bundles and patterns (cf. http://myuipatterns.clevercherry.com/index.php?title=Individualization_Patterns):

- Font Size
 - Font Size – Philips Standard (Default)
 - Font Size – Medium
 - Font Size - Large
 - Font Size – X Large
- Audio Volume
 - Audio Volume - Low
 - Audio Volume - Medium
 - Audio Volume - High
- Audio Speech Output
 - Audio Speech Output – off
 - Audio Speech Output - on
- Audio Speech Speed
 - Audio Speech Speed - normal
 - Audio Speech Speed - slow
- Tonal Audio Output
 - Tonal Audio Output - on
 - Tonal Audio Output - off
- Screen Complexity
 - Screen Complexity – unconstrained

- Screen Complexity – medium
 - Screen Complexity – low
 - Screen Complexity – minimal
- Display Mode
 - Display Mode – mainly text (default)
 - Display Mode – text and graphics
 - Display Mode – mainly graphics
 - Display Mode – graphics only
 - Display Mode – text only
- Navigate and Select
 - Navigate and Select – RC Cursor Navigation
 - Navigate and Select – RC Numeric Navigation
 - Navigate and Select – RC Numeric plus Voice Input
 - Navigate and Select - Voice only
 - Navigate and Select - no input medium found (default)

6.7 Customization Profile

A MyUI customization profile consists of variables which define the branding and style of the application. The variables of the customization profile include the following:

- Colour scheme
defines colour values for specific purposes such as background area, content control background, content control characters, function control background, function control characters, text colour, and title background.
- Font style
defines the font style to be used. In MyUI, this variable is set to Verdana.
- Icon library
defines the location and file names of the icons to be used
- Sound library
defines the location and file names of the sounds to be used

As a default, a MyUI customization profile is available. The application developer can use this customization profile as a standard skin. He can also use it as a starting point to modify and save it under a different name. Customization profiles are created and modified by a developer during the development phase while using the MyUI development toolkit. The desired customization profile is referenced by the variable *skin* in the user interface profile. This variable (*skin*) determines which of the available customization profiles is selected during the stage of user interface parameterization. The selected customization profile cannot be changed for a running application during a single session.

7. User Interface Preparation

User interface preparation denotes the process of selecting the most suitable user interface components and elements for the current situation. In this process four main concepts are involved cf. Figure 22):

- *Abstract Application Interaction Model (AAIM)*
providing the current interaction situations as one of the two major inputs to selecting the appropriate interaction pattern
- *User interface profile*
storing all global user interface variables which reflect the specific requirements related to the individual user and the current device setup (the second major input to interaction pattern selection)
- *Interaction patterns*
providing the knowledge-base (rules) to select the most suitable control and display elements
- *User interface elements*
being called (i.e. required) by the interaction patterns for detailing the higher-level solutions by providing the needed generic primitives

The first two mark the major input of the user interface preparation process. User interface preparation is triggered every time when a new state in the AAIM is entered or when changes have occurred to the user interface profile. The AAIM and the user interface profile are described in earlier sections of this document. Therefore, only interaction patterns and user interface elements are described in detail in the following sections.

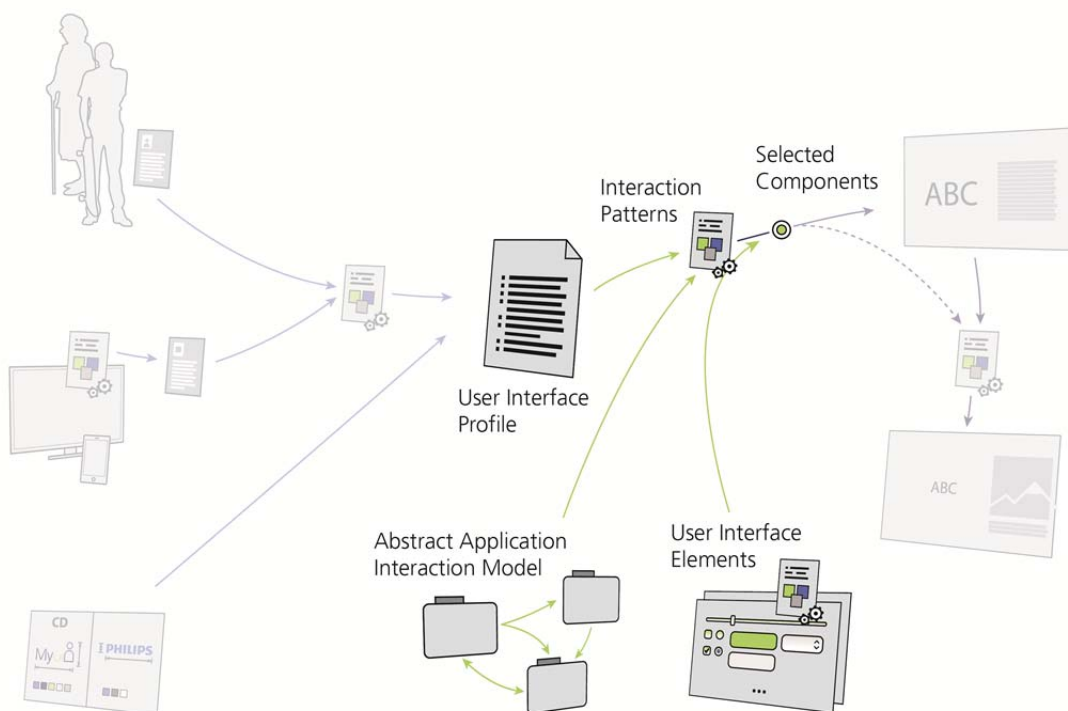


Figure 22 Main concepts and patterns in the MyUI user interface preparation process

7.1 Interaction Patterns

Interaction patterns are the central hub in the process of user interface preparation. They provide the intelligence and knowledge to select the most suitable display and control elements for the current interaction situation as drawn from the abstract application model. For each interaction situation a bundle of interaction design patterns is provided. Depending upon the current values in the user interface profile (as set by the individualization patterns) the most suitable interaction design pattern will be selected from the current bundle in order to cover individual impairments or other constraints. The process of selecting the most suitable interaction patterns is triggered repeatedly during an interaction session with a MyUI application. The following two events trigger a repeated recalculation of all rules of the interaction patterns to activate the best fitting interaction patterns:

- State transitions in the AAIM, i.e. a new interaction situation is active
- Updates of the user interface profile

Interaction design patterns do not have references to the user profile. Their input variables are the current interaction situation drawn from the abstract application model and the relevant user interface profile settings. As an output, interaction patterns call (require) user interface elements (see below) for a more detailed elaboration of certain parts of the solution they provide.

Interaction patterns can have the following relations with other patterns (cf. section 6.3):

- *Substitutes*
grouping two or more interaction patterns to a pattern bundle
- *Requires*
calling user interface elements to detail parts of the solution the interaction pattern describes, e.g. the interaction pattern »Title Bar« requires navigation buttons (user interface element) for the generic navigation functions »Back« and »Home«.
- *uses <variable(s)> as set by*
reflecting the individualized selection of the best-fitting variant of the active interaction pattern bundle, e.g. the user interface profile variable *maxElementsPerScreen* as set by the individualization pattern Screen Complexity determines if a Main Menu (interaction pattern) with sixteen, nine or four menu buttons is displayed.

The MyUI design patterns repository includes bundle descriptions for interaction patterns in the following format:

Table 8 Template for interaction pattern bundles

<ID>	<name of the interaction pattern bundle>
Problem	<describe the interaction situation>
Interaction Situation	<interaction situation>
Parameters	When used in the AAIM, the IS requires the following parameters: <indicate parameters, e.g. content>
Patterns	<indicate the patterns of the bundle>

Single interaction patterns are described as follows:

Table 9 Template for interaction patterns

<ID>	<name of the interaction pattern>
Problem	<describe the interaction situation and relevant conditions of the user interface profile>
Pattern Bundle	<pattern bundle>
Context	IF Interaction situation = <interaction situation> AND <check values of user interface profile variable(s)>
Solution	THEN <Describe the solution in terms of a specific display or control component with concrete feature specifications>
Code reference	<if applicable, provide a reference to the related code to implement the solution>
Diagram	<Illustrate the design solution in a schematic and concise manner>
Examples	<give examples for situations in which the problem can occur and how it can be solved>
Rationale (references)	<explain the principles or rationale behind the pattern and provide references to literature, standards, etc.>
Substitutes	<interaction pattern>
Uses <variable(s)> as set by <pattern bundle>	uses <variable(s)> as set by <pattern bundle>
Requires	<User interface element(s)>

Figure 23 presents an example of an interaction pattern for a 3x3-Main Menu.


Main Menu – 3x3-Tile Style	
3-1-2	Main Menu – 3x3-Tile Style iTV
Problem	<p>Main Menu provides access to all available services/applications of the TV set and might display important status information for some services (e.g. indicate new message for the email service). A reduction to not more than 9 main menu options will be necessary for certain users if:</p> <ul style="list-style-type: none"> • large font sizes are required or • allowed screen complexity reduced or • numeric key pad navigation is activated
Pattern Bundle	Main Menu
Context	<p>If</p> <p>Interaction Situation = <code>SelectService</code></p> <p>AND</p> <p><code>maxElementsPerScreen > 0</code></p> <p>AND</p> <p><code>2,5 < bodyTextFontSize/cellSize < 3,5</code> OR</p> <p><code>numericNavigation = on</code></p> <p>AND NOT <code>displayMode = "text only"</code></p>
Solution	3x3 tile
Code reference	
Diagram	
Examples	
Rationale (references)	
Substitutes:	<p>Main Menu – 4x4-Tile Style</p> <p>Main Menu – 2x2-Tile Style</p> <p>Main Menu – List Style</p> <p>Main Menu – Audio menu</p>
uses <variable(s)> as set by <pattern bundle>	<p>uses</p> <p><code>maxElementsPerScreen</code></p> <p>as set by</p> <p>Screen Complexity</p>

Figure 23 Interaction pattern description (example)

The MyUI design patterns repository currently includes the following interaction pattern bundles and patterns (cf. http://myuipatterns.clevercherry.com/index.php?title=Interaction_Patterns):

- Main Menu (IS=SelectService)
 - Main Menu – 4x4-Tile Style
 - Main Menu – 3x3-Tile Style
 - Main Menu – 2x2-Tile Style
 - Main Menu – List Style
 - Main Menu – Audio menu
- Functions Menu (IS=SelectFunction)
 - Functions Menu - Group of Buttons
- Navigation Menu (IS=SelectGoTo)
 - Navigation Menu - Group of Buttons
- Title Bar (IS=MetaGoTo)
 - Title Bar - iTV
- List with Attributes (IS=SelectItemWithAttributes)

- List with Attributes – 2 Attributes
- List with Attributes – 1 Attribute
- List with Attributes – Tile Style
- List with Attributes – Audio List
- MultiList with Attributes (IS=SelectMultipleItemsWithAttributes)
- TreeStructure (IS=NavigateTree)
- Message Box for Attributes (IS=ProvideFunctionAttributes)
- Message Box for Confirmation (IS=ProvideFunctionConfirmation)
- Item with Attributes (IS=ViewItemWithAttributes)
- Form (IS=EditForm)

7.2 User Interface Elements

User interface elements are the basic building blocks for the current interaction design patterns, e.g. a selection list (interaction pattern) requires option buttons (user interface element). They are quite simple and straight forward. They do not rely on rules like the other pattern types and they are not have variants, i.e. the relation *substitutes* does not apply to user interface elements.

User interface elements can be regarded as a library of common generic controls and display elements which are parameterized by variables of the interaction patterns, e.g. the pattern "action button" defines the general appearance and interaction behaviour of action buttons, the button size, however, is defined by the interaction pattern which calls ("requires") the user interface element. User interface elements can never appear in the MyUI framework without being called (or "required") by a currently selected interaction pattern or another user interface element.

Therefore, user interface elements can have only the following two relations with other patterns (cf. section 6.3):

- *Is required by*
being the main relation for user interface elements which depend heavily on interaction patterns to require them.
- *Requires*
user interface elements can themselves require further user interface elements.

The MyUI design patterns repository currently includes the following user interface elements (cf. http://myuipatterns.clevercherry.com/index.php?title=User_Interface_Elements):

- Action button
- Drop-down button
- Navigation button
- Menu button
- Remote control navigation
- Preview
- Text field
- Text entry field

8. User Interface Generation and Adaptation

User interface generation and adaptation is the last step. The major input to this step is the set of selected user interface components as a result of the user interface preparation (described in section 7). The output of this step is a complete user interface which is consistent with the currently available knowledge about individual user needs and context requirements at any time during the interaction.

A main feature of the MyUI project is the dynamic and system-initiated adaptation of the user interface during run time. Thus, also dynamic changes of the user interface are carried out in this process step. These changes can be considered as a repeated user interface generation with mechanisms to switch from one instance of a user interface to another instance.

In summary, the last step in the MyUI user interface generation and adaptation process includes the following three activities which all take place during run-time:

- *User interface generation*
At the beginning of a new interaction session, a complete user interface is created and rendered on the basis of the selected components (result of user interface preparation).
- *Feedback to the User and Content Management Infrastructure (UCMI)*
During an interaction session, a permanent information stream is transferred to the UCMI in order to provide information about user interactions and user behaviours which are relevant for user profiling.
- *User interface adaptations during use*
When new user interface components and elements have been selected in a repeated user interface preparation process, user interface adaptations are triggered and executed.

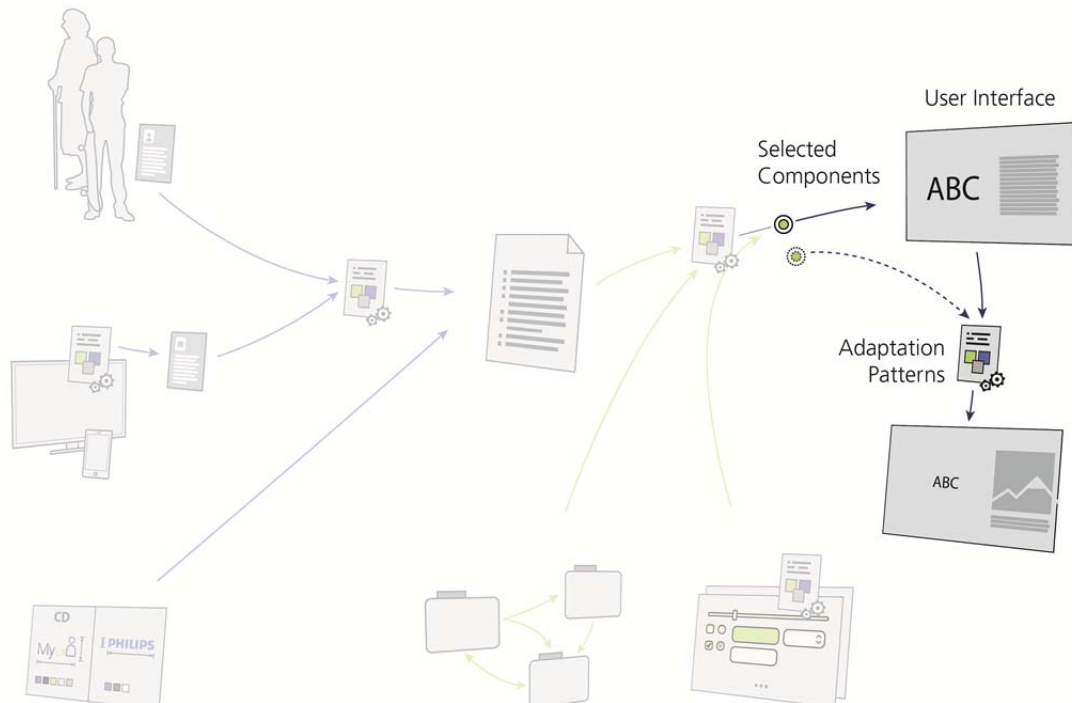


Figure 24 MyUI user interface generation and adaptation from selected components

The main concepts in the MyUI generation and adaptation step include (see Figure 24):

- *Selected components*
The output of the user interface preparation process (cf. section 7) is the main input for the user interface generation. The selected components provide the building blocks for the generated user interface.
- *User interface*
This is the main output of the MyUI user interface generation (see above).
- *Adaptation patterns*
Adaptation patterns are a major concept in the MyUI adaptation process. They control adaptations during use by determining how to move from one instance of a user interface to another. Their main purpose is to ensure high levels of transparency and controllability for the end user in situations of adaptations during use.

8.1 User Interface Generation

A complete user interface is generated by putting together the selected interaction patterns and user interface elements (cf. user interface preparation). User interface generation is triggered at the beginning of a new interaction session with a MyUI application and as soon as changes in the set of selected components have been recognized. Thus, the user interface generation process is triggered by each state transition in the AAIM to create an updated view for the new interaction options. More interesting, however, are cases of repeated user interface generation within one state, i.e. when a new set of user interface components is selected for the current interaction situation. This event is an important trigger for the MyUI Adaptation Engine (see section 8.3).

At the stage of rendering the user interface, the concrete application content is used as an input source to parameterize the user interface widgets, e.g. put specific content options into menu elements and add labels to buttons. In this process, the AAIM (cf. section 4 of this document) provides the bridge between the user interface components and the application data bases. Data acquisition functions connect the application data base with the interaction situations and thus, interaction patterns.

As the main user interaction platform, the MyUI project targets an interactive TV set with the Philips Net TV framework (cf. <http://www.nettv.philips.com>) which relies on CE HTML and is expected to support more advanced web technologies in the future. For target platform support, the MyUI generator creates web-based user interfaces. Involved technologies include HTML, CSS and AJAX/Java Script. The user interface generator is part of the MyUI middleware which is implemented in PHP (see Gacimartin, Hernandez, & Larrabeiti, 2011 for a detailed description of the MyUI middleware). As an output of generic and interaction design patterns, a distinct composition of HTML, CSS and Javascript code is being generated.

8.2 Feedback to the MyUI User and Context Management Infrastructure

During the interaction, the adaptive user interface provides feedback about relevant events in the user interaction to the MyUI user and context management infrastructure (UCMI). This permanent feedback stream shall help the UCOM to obtain reliable information about the user and the context and to refine the current user profile accordingly. Therefore, interaction events which indicate problems of use with the current user interface configuration are of particular interest. These can include the following:

- (repeated) time-out events, i.e. the user does not react in an expected manner within a certain time frame
- repeated go-back events, i.e. the user might not be able to find the path to an intended target state

- repeated invalid user input, i.e. the user might not be able to produce a valid input due to cognitive or motor constraints
- explicit user profile or user interface profile configurations in user interface dialogues (see user-initiated adaptations as described in sections 8.5.6 and 8.5.7),

Further input to the user profile is provided by external sensors which capture relevant user behaviour such as facial expressions, head orientation and eye gaze direction, etc.

Deliverable D1.2 provides a more comprehensive description of sources and mechanisms of capturing, processing and storing information about the user in the MyUI user profile. In order to be able to trigger immediate system reactions to modified user profiles, different client and server sided AJAX components are included into the MyUI adaptation framework.

8.3 Adapting the User Interface during Use

User interface adaptations during use are executed by the MyUI Adaptation Engine. The adaptation engine recognizes mismatches of the user interface components currently displayed at the user interface and the components currently selected by the user interface preparation process for the same interaction situation. This event starts a run-time adaptation process which can be described as a transition from the currently displayed user interface to a new user interface which is composed from the newly selected components. Adaptations are performed by two types of *MyUI Adaptation Patterns*:

- *Adaptation rendering patterns (low-level adaptation patterns)*
specifying the graphical rendering process to smoothly switch from one user interface instance to another, e.g. animated transitions to grow small fonts to bigger fonts
- *Adaptation dialogue patterns (high-level adaptation patterns)*
specifying the user interaction dialogue which takes place around the actual adaptation to make sure that the user is aware of the adaptation and can control the system's adaptation behaviour.

Before describing the MyUI Adaptation Patterns in more detail, we take a look at the benefits and challenges of adaptations during use:

Automatic user interface adaptations during use support the immediate adaptation to newly available information about the user and the context of use. Without requiring the user to change any technical configurations, the user interface can adapt itself immediately to changing user needs and context conditions. Moreover, in contrast to adaptations which occur only between two interaction sessions or at the beginning of a new session (on the basis of the new knowledge from the prior session), adaptations during use can help the user to establish a mental model about the relation of certain interaction events and problems of use and respective user interface adaptations.

On the other hand, however, adaptations during use can also cause problems, such as perception of instability, loss of orientation, control, trust and reliability, etc. In most cases, for example, automatic adaptations will occur in situations when the user experiences problems with the current user interface. In these situations, a changing (i.e. adapting) user interface can even worsen the problem when the changes lead to disorientation. Therefore, the design of transitions from one pattern to another is of major importance for the usability and acceptability of an adaptive system.

8.4 Adaptation Rendering Patterns

Adaptation rendering patterns specify the low-level process of rendering the transition from one user interface instance to the next. This includes several levels and aspects of the user interface. Thus, adaptation rendering patterns can describe the following two different adaptations:

- Interaction pattern → interaction pattern
Switch from one interaction pattern to another interaction pattern of the same bundle.
- Adapt user interface elements
Change display parameters of user interface elements. An example is button labelling which refers to certain user interface profile variables such as font sizes and *displayMode*.

MyUI adaptation rendering patterns make extensive use of animations to support the transition from the old to the new (adapted) user interface. Thus, they support orientation by creating continuity between the user interface before and the user interface after the adaptation. Animated transitions shall draw the end user's attention to the screen areas where the adaptations occur and shall help them to understand that and how the new user interface is a modification of the former user interface. When, for example, an increased font size results in hiding menu options which were directly available before the adaptation, an animation can communicate to the user that the hidden options are now available via the »more« button.

Adaptation rendering patterns can be referred to by interaction patterns or interaction pattern bundles and by user interface elements. Adaptation rendering patterns are triggered by the higher-level adaptation dialogue patterns as described in section 8.5. The only relation applicable to adaptation rendering patterns is the *is required by* relation: (cf. section 6.3):

- *Is required by*
being the only relation for adaptation rendering patterns which are called by an ordered triple of one adaptation dialogue pattern and two interaction patterns of the same bundle or a pair of an adaptation dialogue pattern and a user interface element.

8.5 Adaptation Dialogue Patterns

Adaptation dialogue patterns specify the dialogue between the user interface and the MyUI end user in the course of an adaptation. This dialogue can include the presentation of specific forms of notifications about an adaptation and interaction options for the end user to influence the system's adaptation behavior.

Adaptation dialogue patterns always include adaptation rendering patterns which specify the concrete process of switching from one instance of the user interface to another. Adaptation dialogue patterns can have the following relations with other patterns:

- *Substitutes*
grouping adaptation dialogue patterns to a pattern bundle.
- *Requires*
calling adaptation rendering patterns to detail the lower-level transitions. This relation applies to an ordered triple of one adaptation dialogue pattern and two interaction patterns of the same bundle or an ordered pair of one adaptation dialogue pattern and a user interface element requiring an adaptation rendering pattern.

In MyUI, two types of adaptation dialogue patterns are distinguished:

1. *System-initiated adaptation dialogue patterns* (see sections 8.5.1 to 8.5.4)
triggered by the system in the course of an automatic user interface adaptation
2. *User-initiated adaptation dialogue patterns* (see sections 8.5.6 and 8.5.7)
describing dialogues which enable the end user to actively trigger modifications of the user interface

System-initiated adaptation dialogue patterns aim at increasing the usability and acceptability of self-adaptive user interfaces by ...

- ... increasing the *transparency* of automatic adaptations,
i.e. making adaptations clear and understandable to the end user and
- ... increasing the *controllability* of automatic adaptations,
i.e. providing the end user the opportunity to influence the system's adaptation behaviour.

In terms of transparency, adaptation dialogue patterns can reinforce the effect of the adaptation rendering patterns (e.g. animations) by providing explicit hints or indications that an adaptation has occurred or will occur. However, additional information displays for on-going or recent adaptations will load the screen and might result in attention problems, distraction and mental overload on the end user's side.

The controllability of automatic adaptations can be supported by the following mechanisms:

- Requesting an explicit end user confirmation before an adaptation is executed
- Requesting an explicit end user confirmation after an adaptation is executed
- Providing an undo function after the adaptation has been executed (implicit)

Explicit confirmations can be assumed to result in higher control levels than the implicit option of an undo function. On the other hand, explicit confirmations increase the interaction effort by adding an additional interaction step which might also lead to additional interaction problems.

In summary, the design of the most suitable adaptation dialogue pattern can be understood as optimizing the trade-off between transparency and information conciseness as well as controllability and efficiency of interaction.

The following sub-sections describe the system-initiated adaptation dialogue patterns of the MyUI pattern repository (cf. http://myuipatterns.clevercherry.com/index.php?title=Adaptation_Patterns).

8.5.1 Explicit Confirmation before Adaptation

Before performing the adaptation the system requests the end user to explicitly accept or reject the adaptation. The user's decision is supported by providing a preview of the effect of the adaptation. If the user rejects the adaptation, the dialogue box is closed and the current user interface is not changed. If the user accepts the adaptation or if the system receives no user input (time-out), the dialogue box is closed and the adaptation is carried out. Moreover, the MyUI user and context management infrastructure is informed about the user's decision (feedback to user profiling). Figure 25 presents the interaction sequence of the pattern. In Figure 26 the dialogue box for the explicit confirmation is outlined.

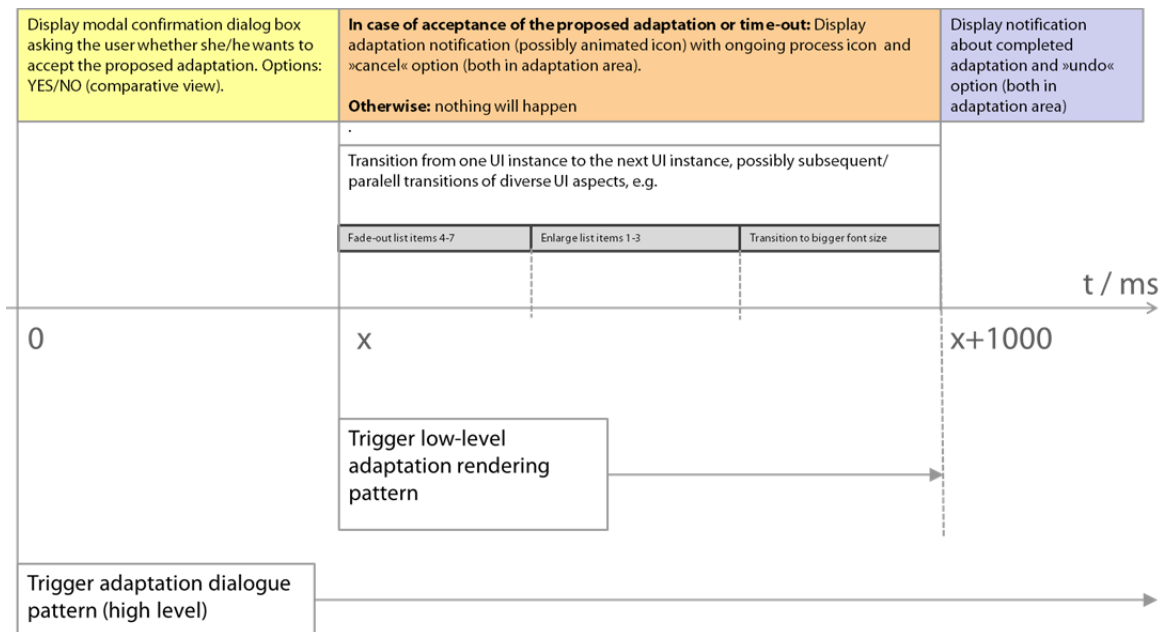


Figure 25 Adaptation dialogue pattern »Explicit Confirmation before Adaptation«: sequence diagram

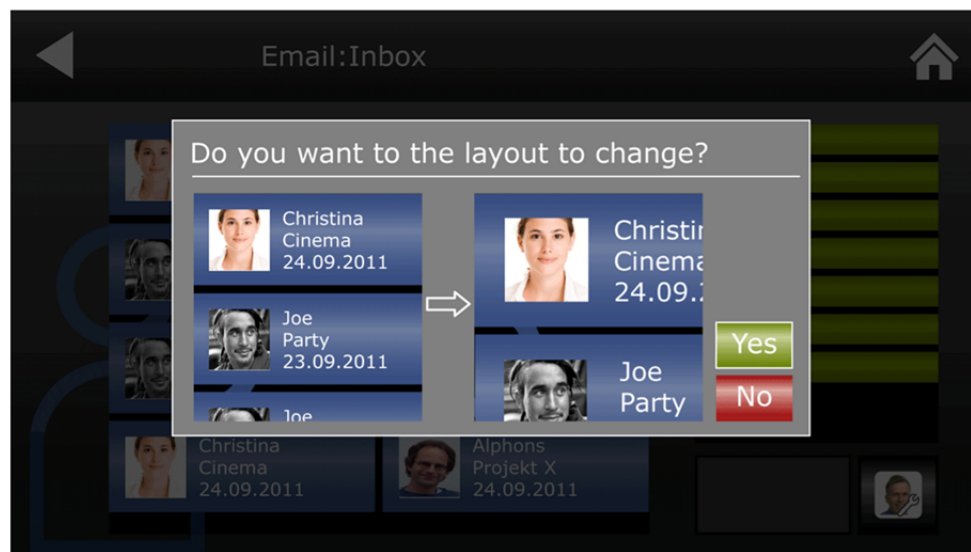


Figure 26 Adaptation dialogue pattern »Explicit Confirmation before Adaptation«: Dialogue box with adaptation preview (user interface concept sketch)

8.5.2 Explicit Confirmation after Adaptation

The adaptation is triggered and performed automatically. When the adaptation with all adaptation rendering patterns has been finished, a dialogue box is displayed which asks the end user if the changes shall be kept or undone. If the user rejects the adaptation is undone. If the user accepts the adaptation or if a time-out event is registered, the adaptation is kept. In any case, the end user's decision in the dialogue box is fed back to the user and context management infrastructure to support refining the user profile. Figure 27 presents the interaction sequence of the pattern. The dialogue box is similar to the box of the »Explicit Confirmation before Adaptation« pattern. The prompt in the dialogue box is: »The layout has been changed. Do you want to keep the changes?«.

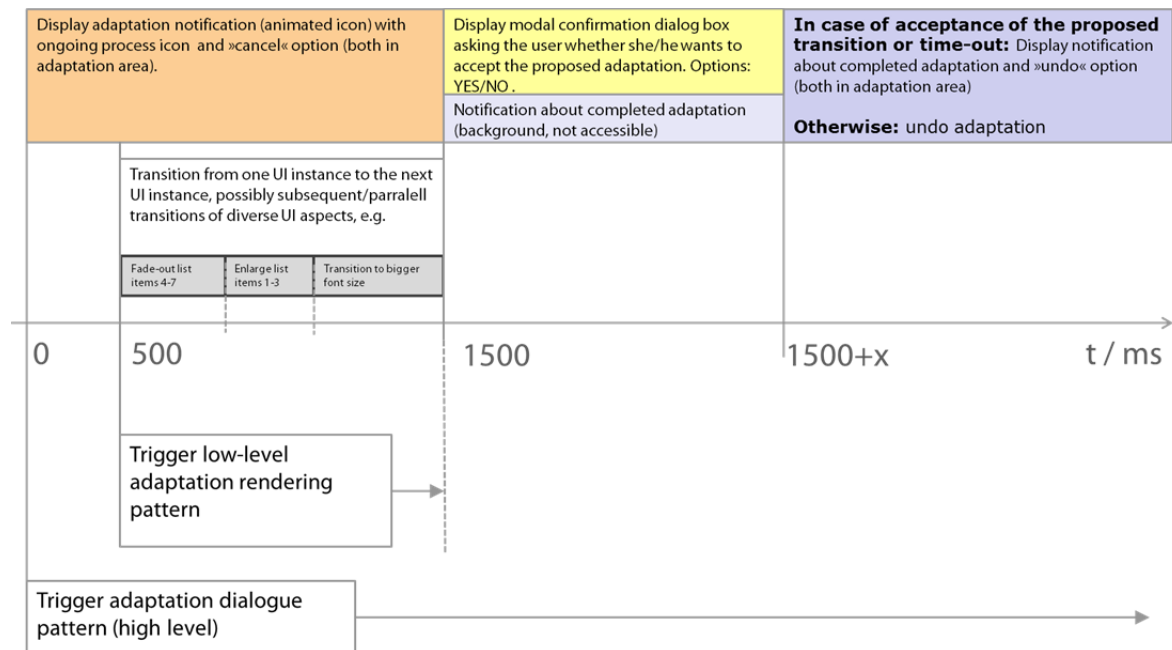


Figure 27 Adaptation dialogue pattern »Explicit Confirmation after Adaptation«: sequence diagram

8.5.3 Automatic Adaptation with Implicit Confirmation

The adaptation is triggered and performed automatically. While rendering the adaptation, the system provides a notification about the on-going adaptation in the adaptation area. When the adaptation is finished the end user can undo the adaptation via a button in the adaptation area. Moreover, the adaptation area provides a permanently available access to the user interface profile and the user profile. In Figure 28, the interaction sequence of the pattern is presented. Figure 29 shows a sequence of three screens before, during and after an automatic adaptation with implicit confirmation. The implicit adaptation confirmation dialogue is taking place in the adaptation area at the bottom right of the screen. The user's behaviour in this implicit dialogue is fed back to the user and context management infrastructure, i.e. undoing and accepting the adaptation is interpreted in terms of consistency or mismatch between the displayed user interface and individual user needs.

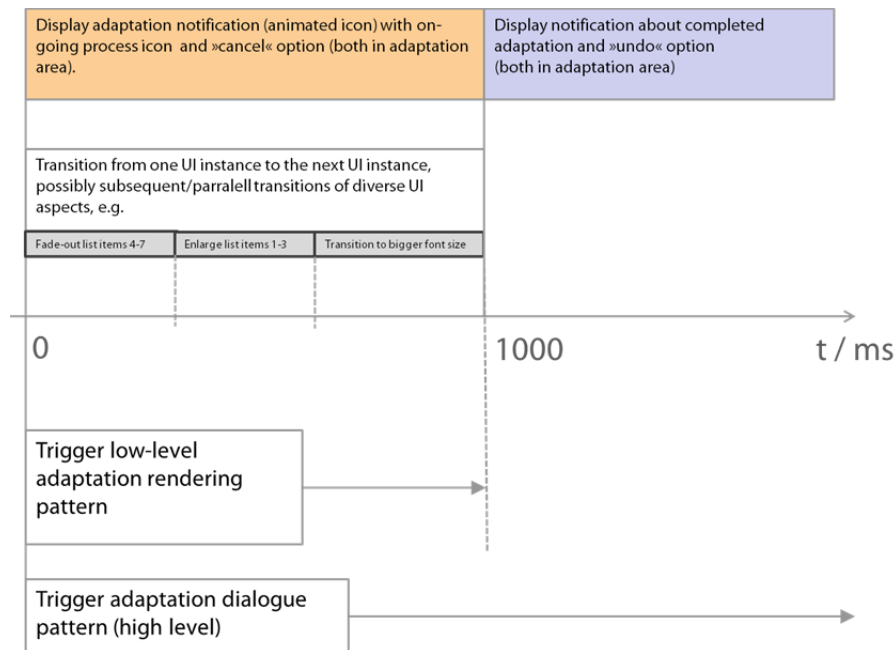


Figure 28 Adaptation dialogue pattern »Automatic Adaptation with implicit Confirmation«: sequence diagram

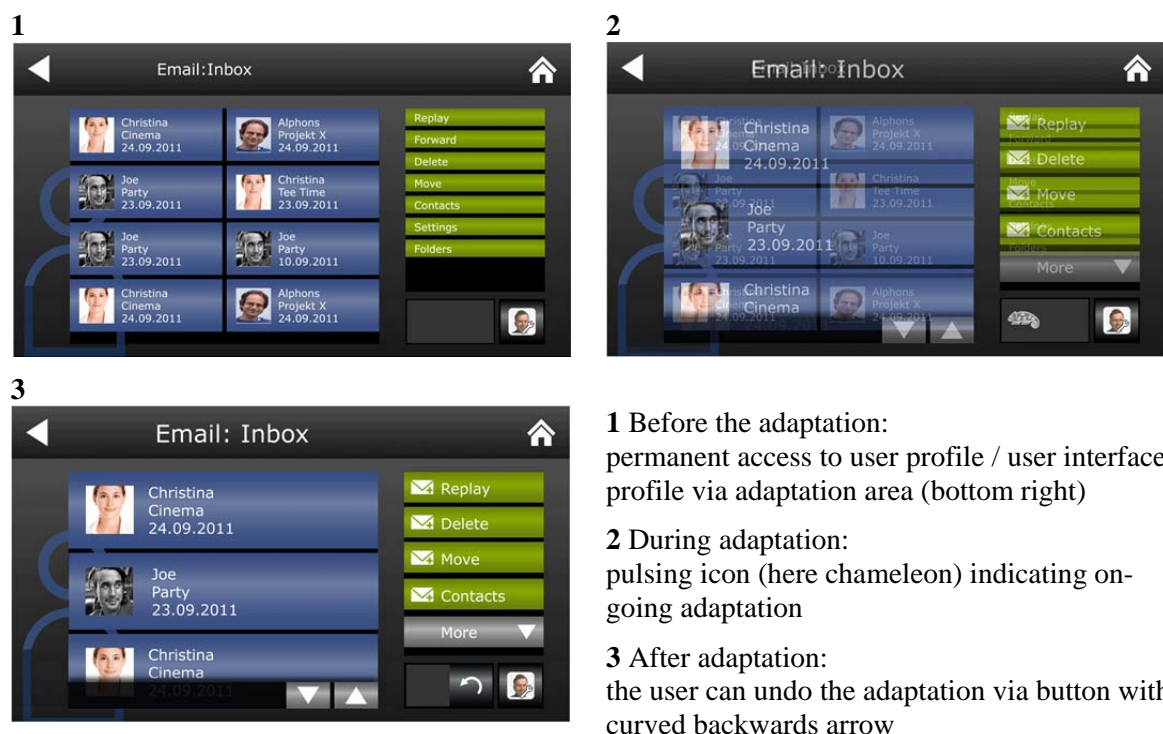


Figure 29 Screen sequence during an automatic adaptation with implicit confirmation

8.5.4 Automatic Adaptation without Adaptation Dialogue

This adaptation pattern can be regarded as a baseline condition. Whereas, all other adaptation dialogue patterns call an adaptation rendering pattern in the course of a dialogue between the user interface and the end user, this pattern does not have a dedicated adaptation dialogue. It triggers just the lower level adaptation rendering pattern without any notifications or confirmation prompts. It neither offers any undo option to the user. Therefore, it is assumed to offer the lowest levels of

transparency and controllability compared to the other adaptation dialogue patterns.

8.5.5 Selecting the most appropriate system-initiated adaptation dialogue pattern

The MyUI Adaptation Engine manages and controls all run-time adaptations. This includes also the selection of the most suitable adaptation dialogue pattern for a current situation. From a technical point of view, the adaptation engine is currently able to base its decision on the following input parameters:

- *Severity of the adaptation*
Every possible adaptation, i.e. each pair of interaction patterns of a common bundle and each adaptable parameter of a user interface element, receives a severity rating on a three-point-scale (major, medium, minor change). The severity of an adaptation can then be used as criteria for selecting an adaptation dialogue pattern, e.g. minor changes such as adapting the contrast to the ambient lighting conditions will not need a special notification to the users whereas significant changes such as switching from manual to voice control will require more apparent and explicit notification and confirmation dialogues.
- *Individual characteristics*
Personal traits as stored in the user profile can be used to hypothesize which adaptation dialogue will be preferred by an individual end user. In user studies carried out by Fraunhofer IAO, we found out that lower levels of ICT literacy are often associated with a personal preference for more confirmations and explicit notifications in user interfaces.

The technical implementation of this selection process is unproblematic and already done in MyUI. However, the definition of a set of selection rules which make sure that the selected adaptation dialogue patterns establish the highest possible levels of transparency and controllability is quite a challenge. In order to define a valid selection rule and put it on an empirical basis, user studies will be carried out in January and February 2012. The main objectives and test issues can be summarized as follows:

- Which adaptation dialogue patterns ...
 - Explicit confirmation before adaptation
 - Explicit confirmation after adaptation
 - Automatic adaptation with implicit confirmation
 - Automatic adaptation without adaptation dialogue
- ... are best suitable...
 - in terms of transparency,
 - subjective level of control,
 - efficiency of use and
 - user acceptance and preference
- ... under different adaptation conditions?
 - Subjective benefit: low vs. high (e.g. increase font size for user with low vision)
 - Subjective disadvantage: low vs. high (e.g. increased font size results in hiding options of personal interest)
 - Case-specific or general (personal) preference?

A detailed description of the experimental setup and the results will be given in the next report of WP 5.1.

8.5.6 User-initiated Adaptation via User Profile

Sections 8.5.1 to 8.5.4 describe adaptation dialogue patterns for cases in which the adaptation is initiated by the adaptive system when user profile updates lead to respective updates in the user interface profile and the set of selected user interface components. Besides system- initiated adaptations, the MyUI adaptation framework provides also opportunities for user-initiated adaptations.

The first option for end users to initiate modifications on the user interface is to change their user profiles. In favour of high levels of privacy and transparency, the MyUI framework offers a permanently available access to the user profile where the end user can get an overview of all user-related information stored in the MyUI system. The user profile includes variables such as *visualAcuity*, *attention*, *hearing*, *fingerPrecision*, etc. (cf. Table 4 in section 6.4). These variables represent the individual user's capabilities and constraints. The end user can directly change all user profile parameters.

If, for instance, a user increases the value of the variable *visualAcuity* in her user profile, this change is immediately transmitted to the Context Manager which registers and handles the change as an incoming sensor event. This leads to resuming the user interface parameterization process which results in updating the user interface profile and, finally, to an adaptation of the user interface. The process of a user-initiated adaptation via user profile can be subdivided into the following steps (see Figure 30):

1. The end user changes a value in the user interface representation of the user profile e.g. *visualAcuity*.
2. The context manager receives a sensor event and updates the user profile variables accordingly.
3. This change restarts the user interface parameterization process automatically which leads to an update of the user interface profile.
4. The changes in the user interface profile trigger the process steps of user interface preparation and user interface generation and adaption which result in an adapted user interface.

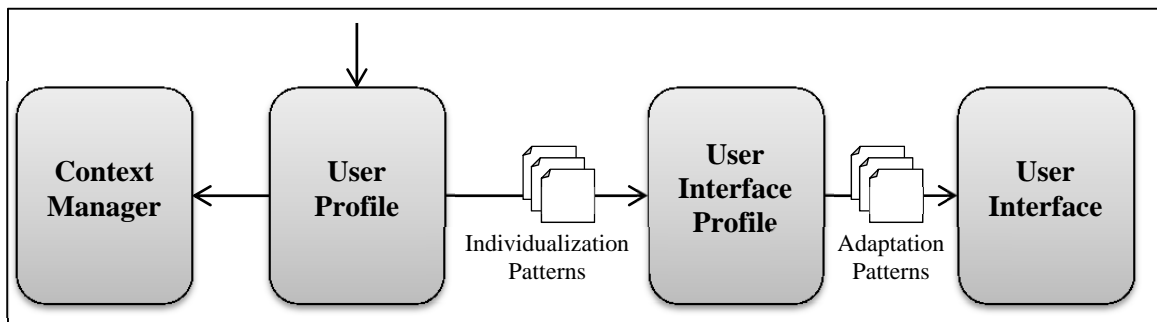


Figure 30 User-initiated adaptation via user profile

8.5.7 User-initiated Adaptation via User Interface Profile

The second option for end users to initiate user interface modifications is changing the user interface settings as stored in the user interface profile. The user interface profile is permanently accessible for end users of MyUI applications. In the user interface profile view the user can have an overview of all user interface profile variables such as *bodyTextFontSize*, *audioOutputVolume*, *numericNavigation*, *maxElementsPerScreen*, etc. (cf. Table 5 in section 6.5) and change them directly.

User-initiated changes in the user interface profile start a complex process. The major part of this process aims at assuring consistency between the modified user interface profile and the current

user profile. The underlying assumption is that end users change user interface settings in a way that they better suit their individual needs and preferences. As a consequence, user interface profile modifications can be used in order to refine the user profile. And under the above assumption, this refinement can be achieved by updating the user profile so that it is consistent to the new user interface profile. However, user-initiated user interface profile changes can lead to conflicts which make consistent user and user interface profiles impossible. Moreover, a unique solution to resolve inconsistencies will not exist in some constellations of the user interface profile. In these cases, clarification and/or disambiguation dialogues will be needed.

The process of a user-initiated adaptation via user interface profile can be described as follows (see Figure 31):

1. The end user changes a value in the user interface profile.
2. A reverse application of the individualization patterns identifies an updated user profile which is consistent with the newly modified user interface profile and which requires the least user profile changes.
3. If this is not possible, the system suggests further user interface profile modifications to the end user. This interactive process ends with a user-confirmed user interface profile for which a consistent and unambiguous user profile can be created.
4. The suggested user profile updates are communicated to the context manager where the user profile is updated accordingly.
5. The changes in the user interface profile trigger the process steps of user interface preparation and user interface generation and adaption which result in an adapted user interface.

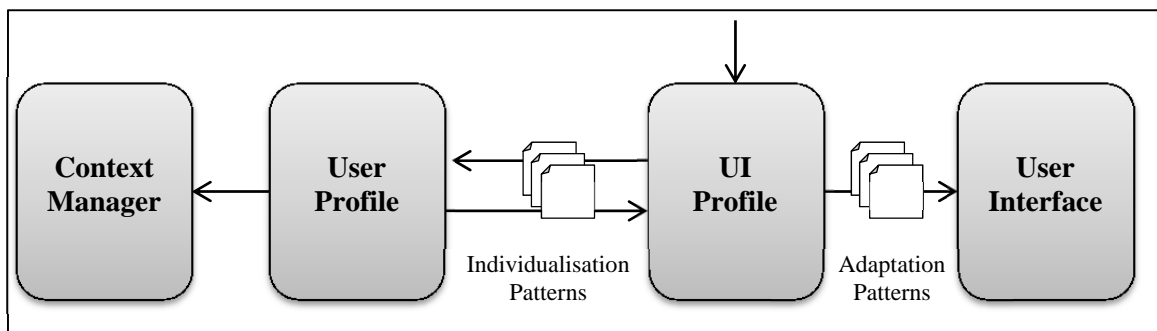


Figure 31 User-initiated adaptation via user interface profile

References

- Borchers, J. O. (2000). CHI meets PLoP: An interaction patterns workshop (at ChiliPLoP'99 Conference on Pattern Languages of Programming, Wickenburg, AZ, March 16-19, 1999). *SIGCHI Bulletin*, 32(1): 9-12, January 2000.
- Borchers, J. O. (2001). A pattern approach to interactive design. Chichester, UK: John Wiley & Sons Ltd.
- Brusilovsky, P. & Maybury, M.T. (2002): From Adaptive Hypermedia To The Adaptive Web, in: Communications of the ACM. Volume 45, Number 5 (2002): 31–33.
- Dieterich, H., Malinowski, U., Kühne, T. & Schneider-Hufschmidt, M. (1993). State of the Art in Adaptive User Interfaces. In: M. Schneider-Hufschmidt, T. Kühne & U. Malinowski (eds.): Adaptive User Interfaces: Principles and practice. Amsterdam: North-Holland. pp. 13–48.
- Findlater, L. & Gajos, K.Z. (2009). Design Space and Evaluation Challenges of Adaptive Graphical User Interfaces. *AI Magazine* 30(4).
- Gacimartin, C., Hernandez, J.A. & Larrabeiti, D. (2011). A middleware architecture for designing TV-based adapted applications for the elderly. In: J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, Berlin: Springer-Verlag. pp. 443–449.
- Gajos, K.Z. & Weld, D.S. (2004). "SUPPLE: Automatically Generating User Interfaces". IUI'04. Ed. ACM. IUI. 2004. 93–100.
- Gajos, K.Z., Wobbrock, J.O. & Weld, D.S. (2008). Improving the Performance of Motor-Impaired Users with Automatically-Generated, Ability-Based Interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'08), 1257–1266. New York: Association for Computing Machinery.
- Gamma, E., Helm, R., Johnson R. & Vlissides J. (1994). Design Patterns – elements of Reusable Object-Oriented Software. Addison-Wesley Longman, Amsterdam.
- Harper, R., Rodden, T., Rogers, Y. & Sellen, A. (2008). Being Human: Human-Computer Interaction in 2020. Microsoft Research Ltd.
- Hartmann, M., Schreiber, D. & Mühlhäuser, M. (2009). "AUGUR: Providing Context-Aware Interaction Support". EICS'09. Ed. ACM. EICS. 2009. 123–131.
- Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender Systems: An Introduction. Cambridge University Press.
- Jason, B., Calitz, A.P., & Greyling, J.H. (2010). The evaluation of an adaptive user interface model. Proceedings of SAICSIT Conference 2010, pp.132-143.
- Jeckle, M., Rupp, C., Hahn, J., Zengler, B. & Queins, S. (2004). UML 2 glasklar. Hanser Verlag, 2004.
- Krulwich, B. (1997). "LIFESTYLE FINDER: Intelligent User Profiling Using Large-Scale Demographic Data". *AI Magazine*. Volume 18, Number 2 (1997): 37–45.
- Krzywicki, A., Wobcke, W. & Wong, A. (2010). An Adaptive Calendar Assistant Using Pattern Mining for User Preference Modelling. ACM New York, NY, USA.
- Leuteritz, J.-P., Widloirer, H., Mourouzis, A., Panou, M., Antona, M., Leonidis, A. (2009). Development of Open Platform Based Adaptive HCI Concepts for Elderly Users. In: Stephanidis, C. (ed.) Proceedings of 13th International Conference on Human-Computer Interaction (HCI International 2009). Berlin: Springer Verlag.

- Moustakas, K., Kaklanis, N., Tzovaras, D., Peissner, M., Lawo, M., & Hahn, V. (2010). User Model Interoperability Requirements. Public deliverable D1.6.4 of the European project VERITAS. <http://veritas-project.eu>
- MyUI Design Patterns Repository (2011). <http://myuipatterns.clevercherry.com>
- Nielsen, J. (1993) Usability Engineering, Cambridge MA: Academic Press.
- Object Management Group (2010). Unified Modeling Language (OMG UML) Superstructure, Version 2.3, May 2010, chapter 15 on “state machines”.
- Orwant, J. (1994). Heterogeneous learning in the doppelgänger user modelling system. *User Modeling and User-Adapted Interaction*, 4(2):107–130, June 1994.
- Paramythis, A., Stephanidis, C. and Totter, A. (2001). A modular approach to the evaluation of Adaptive User Interfaces. In Chin, D., Weibelzahl, S. and Weber, G. (Eds.) *Adaptive evolutionary information systems*. Sonthofen, Germany, 9-24.
- Peissner, M., Schuller, A. & Spath, D. (2011). A design patterns approach to adaptive user interfaces for users with special needs. In: J.A. Jacko (Ed.): *Human-Computer Interaction, Part I, HCII 2011, LNCS 6761*, Berlin: Springer-Verlag, pp. 268–277.
- Peng, X., & Silver, D. L. (2007). Interface Adaptation Based on User Expectation. *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, S.264-269, May 21-23.
- Philips. <http://www.nettv.philips.com>
- Ringbauer, B., Peissner, M., & Gemou, M. (2007). From “design for all” towards “design for one” – A modular user interface approach. In: *Proceedings of the 4th Int'l Conference on Universal Access in Human-Computer Interaction (UAHCI '07)*. Berlin: Springer-Verlag, pp. 517–526.
- Takacs, B., Simon, L., & Peissner, M. (2011). Sensing User Needs: Recognition Technologies and User Models for Adaptive User Interfaces. In: J.A. Jacko (Ed.): *Human-Computer Interaction, Part I, HCII 2011, LNCS 6761*, Berlin: Springer-Verlag, pp. 498–506.
- Thorrtas, C. G. & Krogsoeter, M. (1993). "An Adaptive Environment of the User Interface of Excel". *Intelligent User Interfaces '93*. 123–130.
- Totterdell, P. & Rautenbach, P. (1990). Adaptation as a Problem of Design. In D.Browne, P. Totterdell, & M. Norman (Eds.), *Adaptive User Interfaces* (pp. 61-84). London: Academic Press.
- Weber, G. & Specht, M. (1997). "User Modeling and Navigation Support in WWW-Based Tutoring Systems". In: A. Jameson, C. Paris and C. Tasso (eds.): *Proceedings of 6th International Conference on User Modeling*. Wien: Springer Wien New York, pp. 289-300.
- Weibelzahl, S. (2002). *Evaluation of Adaptive Systems*. Dissertation. Trier: University of Trier.