Chapter 9
Building an Adaptive Multimodal Framework for Resource Constrained Systems

Carlos Duarte, Daniel Costa, Pedro Feiteira, and David Costa

Abstract Multimodal adaptive systems have typically been resource consuming systems, due to the requirements of processing recognition based modalities, and computing and applying adaptations based on users and context properties. In this chapter, we describe how we were able to design and implement an adaptive multimodal system, capable of performing in a resource constrained environment such as a Set-top Box. The presented approach endows standard non-adaptive, non-multimodal applications with adaptive multimodal capabilities, with limited extra effort demanded of their developers. Our approach has been deployed for Web applications, although it is applicable to other application environments. This chapter details the application interface interpretation, multimodal fusion and multimodal fission components of our framework.

9.1 Introduction

The current trend shows computational devices moving into the living room. People use tablets and smartphones while they watch TV, for tweeting about what their watching, or to find additional information about the show [1]. Set-top boxes and connected TVs are making this possible even without the additional devices. This trend also means the number of potential users is increasing, and with it the diversity of users' abilities, characteristics and technical knowledge will also increase. Combine this increasingly diverse user population with their lack of knowledge about the use of such applications, and the lack of traditional input devices (i.e. mouse and keyboard) in the living room setting, for which most web applications (that are now being made available in TV sets) have been built, and it is possible to envision a high resistance to their adoption. A solution to such problem requires

C. Duarte (✉)
Department of Informatics, University of Lisbon, Lisboa, Portugal
e-mail: cad@di.fc.ul.pt

D. Costa • P. Feiteira • D. Costa
Faculty of Sciences, University of Lisbon, Campo Grande, 1749-016 Lisboa, Portugal
e-mail: dancosta@di.fc.ul.pt; pfeiteira@di.fc.ul.pt; dcosta@lasige.di.fc.ul.pt

P. Biswas et al. (eds.), *A Multimodal End-2-End Approach to Accessible Computing*,
Human–Computer Interaction Series, DOI 10.1007/978-1-4471-6708-2_9

an approach that can support natural interaction modalities, with the capability to interpret inputs from multiple modalities, and the capability to adapt the presentation of current and future web applications to the abilities and skills of the user and to the context of use, characterized by the existing devices and the surrounding environment (which takes particular importance in the living room scenario).

Multimodal input interpretation (involving recognition of speech, gestures, and possibly other modalities), and distribution and adaptation of output rendering over different modalities, are computationally expensive operations. On the other hand, set-top boxes have limited processing power, so some constraint on the multimodal operations has to be exercised. Still, their processing power is increasing, and the differences to other living room devices, like gaming consoles, can be expected to decrease, and even possible merge into one device, which opens new perspectives for the future. However, presently, limitations still have to be built into the adaptation process, as will be presented in the corresponding sections of this chapter. Nevertheless, the features that motivate this development have to be retained, or else the platforms will not be adopted by end-users.

Platform and service providers are not expected to deliver a platform with adaptive multimodal characteristics. The work on these platforms have been conducted almost exclusively in the realm of academia and research institutions, although in the recent years, natural interaction modalities have been making their way into the living room, promoted essentially by gaming consoles. As such, to further their adoption, and to bring the benefits of adaptation and multimodal interaction to all the strata of the population, the solution must be a framework capable of interfacing between the user and a standard application. A service provider, will then benefit from having a framework that can translate multimodal inputs into something that the application can process, and that additionally is capable to adapt the application's interface in a way that best suits the user. The end-user benefits from having the adapted presentation and from being able to interact naturally. This chapter presents that framework, focusing on the mechanisms that enable adaptive multimodal fusion and fission.

The next section will give a brief overview of the framework's architecture. The following section describes how the framework and applications communicate user interface (UI) information between them. The following two sections describe the mechanisms that endow the framework with adaptive multimodal fusion and fission capabilities. The final section concludes this chapter and presents an outlook for future developments.

9.2 Architectural Overview

The GUIDE[1] project has developed a framework, that sits between the interaction devices employed by users and applications deployed by service providers, endowing applications with adaptive multimodal interaction capabilities.

[1] http://www.guide-project.eu/

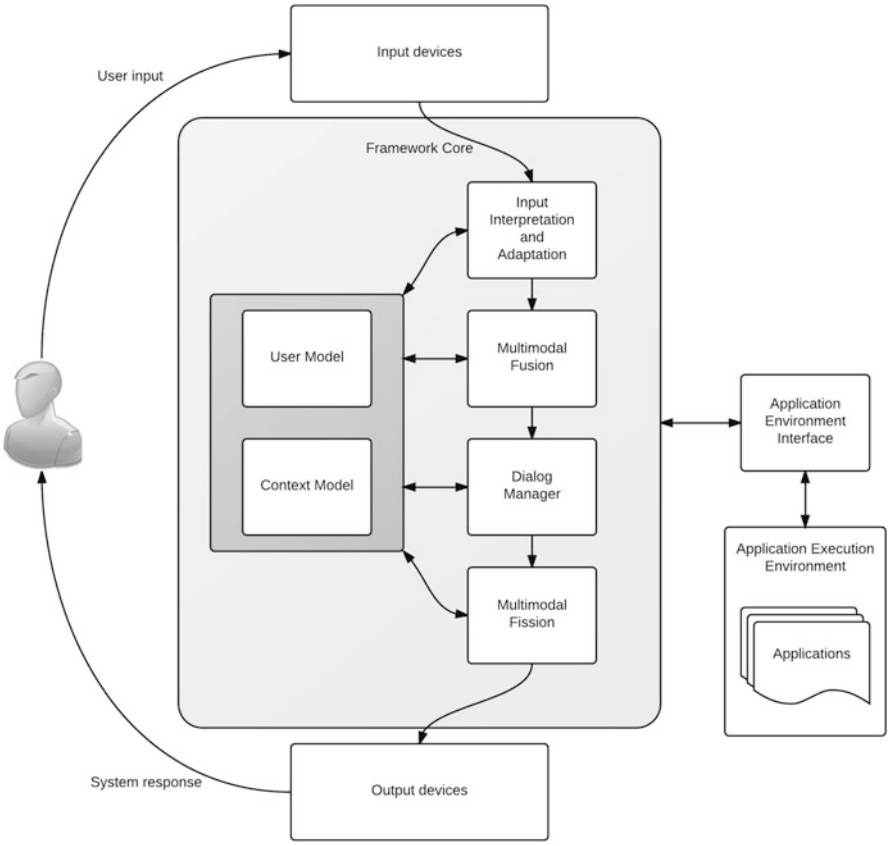

Fig. 9.1 Architectural overview of the GUIDE framework

Figure 9.1 presents an architectural overview of the GUIDE framework and communication system.

Applications are executed in environments (e.g. web applications in browsers, java applications in java runtime environments). The framework abstracts this into the concept of Application Execution Environment (AEE). For each AEE, an Application Environment Interface (AEI) needs to be provided. The AEI is responsible for managing the communication between applications and framework, which includes translating the application's UI into a representation that is understood by the framework and exchanging events between application and framework. Based on the application's abstract representation the framework performs runtime adaptation of the interface elements (such as adjusting text-size, font-size, color, and contrast) and uses different input and output modalities for interaction with the user.

The framework core specification defines two groups of components. The first set of components implements multimodal adaptation algorithms for processing and managing input and output data as well as to adapt parameters across modalities for

a given user profile. These include: the "Input Adaptation" module for filtering and manipulating a sequence of continuous user input data such as cursor positions from a computer mouse; the "Fusion Module" that is responsible for interpreting inputs from different input devices into meaningful commands for a given application; the "Dialog Manager" manages changes in the application's state and manages dialogs between user and framework; and the "Fission Module" that is responsible for preparing and coordinating the multimodal rendering of content to the user. A second set of components manages context information and user profile data. The Context Model stores and manages context events generated by different context sources and implements rule-based logic for reasoning on context data for given situational changes. The User Model manages a set of sample profiles derived by a user initialization application at runtime.

Further details of the communication mechanisms of the framework and of framework components are presented in other chapters of this book. Still, it is important to highlight two factors: application independence and platform independence. For the framework to be adopted by service providers it must be able to process applications which have not been programmed specifically for it. For the framework to be adopted by platform manufacturers it must be able to operate in platforms which have not been designed specifically for it. In order to support these requirements, AEI will have to be provided for each AEE. In the scope of the GUIDE project, on AEI has been developed targeting one AEE. The AEE selected was the web browser, and the resulting AEI is called the Web Browser Interface (WBI). Additionally, an application independent UI specification format has been adopted in the framework, which will be presented in the next section.

9.3 Interfacing with Applications

In order for an adaptive system to adapt a user interface or to apply the interpreted commands from combinations of different inputs, it needs to have knowledge about the application's user interface. One the GUIDE framework's goals is to be compatible with any type of application. As it isn't feasible to develop an instance of the core for each application type (and associated development language), we needed a standard UI description language (UIDL) to interface between framework and application.

Further, as has been discussed above, framework requirements specify that application designers and developers should have little effort to make their applications compatible with the framework. At the same time, the extraction of the UI representation should be possible without requiring much processing power due to the constraints imposed by set-top box environment. To meet these requirements each AEI must have a component, named User Interface Extraction Component (UIREC), that is responsible for extracting the application's UI representation.

The following sections describe the alternatives and choices we have faced during the development of this component for the WBI.

9.3.1 User Interface Description Languages

As mentioned before, it is a requirement of the GUIDE framework to analyze, choose and adhere to one of existing abstract UI representation language standards. In order to ensure a complete separation of application logic from its presentation, the GUIDE Framework should support a standard interface vocabulary that abstracts the semantics and intents of an application from its concrete user interface implementation. Thus a design consideration in GUIDE is to evaluate and select from the plethora of existing abstract UI description standards, one that meets its abstract UI description needs but at the same time, also requires minimal effort in implementation and re-use. Table 9.1 shows the considered UIDLs.

XForms cannot cover all the requirements for GUIDE as it is incapable of describing more specific properties or attributes of certain elements (Buttons, Images, etc.) such as position values, size, color or other style properties. Although much more complete than XForms, UsiXML fails in giving the location of objects which is an important property for Fusion and User Model components in the

Table 9.1 User interface description languages considered in the GUIDE project

UIDL name	Description
XForms (http://www.w3.org/MarkUp/Forms/)	XML application that represents the next generation of forms for the web, and has introduced the use of abstractions to address new heterogeneous environments. When comparing XForms with HTML Forms, the main difference, apart from XForms being in XML, is the separation of the data, from the markup of the controls
UsiXML (http://www.usixml.eu/)	Standing for USer Interface eXtensible Markup Language, it is an XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces
XIML (http://www.ximl.org/)	It is an extensible XML-based specification language for multiple facets of multiple models in a model-based approach, developed by a forum headed by RedWhale software. It was introduced as a solution that enables a framework for the definition and interrelation of interaction data items
UIML (https://www.oasis-open.org/)	The User Interface Markup Language, is an example of a language that has addressed the multi-device interface issue. It is an XML-compliant language that supports a declarative description of a user interface in a device-independent manner. In UIML, a user interface is a set of interface elements with which the user interacts. These elements may be organized differently for the different categories of users and types of appliances

GUIDE Core. Although the tag and property names are a bit sloppy they are indeed complete, but the main drawback connected to XIML is the fact that, differently from the majority of the other User Interface description languages, it is developed within a software company, and therefore its use is protected by copyright. UIML specification does not define property names. This is a powerful concept, because it allows UIML to be extensible: one can define whatever property names are appropriate for a particular element of the UI. For example, *color* might be a useful property for a button, while *text-size* might be appropriate for a label. This flexibility allows us to define all the properties needed for all GUIDE components (Input Adaptation, Fusion, Fission, Dialogue Manager, User Model, etc.). Additionally, they might be used to represent the information developers might provide using WAI-ARIA [2] markup tags. UIML seems to be the most complete and flexible UI representation, therefore it was chosen as the standard language to be used as the interface language between the framework and the applications.

Other adaptive multimodal systems such as EGOKI [3] also use this UIDL. However, our approach is different as the framework is the one generating the UIML automatically based only on the UI elements, discarding the logical specification of the application.

9.3.2 Implementation Alternatives

Before the UIREC was made part of the WBI different approaches on where and when this extraction was to be done were considered.

One of the first approaches was to do it in design time, i.e., when developers finished their application they would use a tool to extract a UIML file describing the entire application. However, this approach was discarded because different developers use different implementation methods. For instance, some developers use separate files for different application states (e.g. one HTML file for each state) and other developers use one single file. Developing a parser for this scenario would be a task of extreme complexity.

The next approach was to integrate this component in the GUIDE core serving as an Application Model that would derive and extract automatically the states and UI representations. However, this suffered from the same limitations and, additionally, this approach would require a change inside the GUIDE core every time a new application execution platform, and thus application language, is integrated.

The final approach is to make this extraction state by state and outside the core. This approach has the advantage of delegating to the AEI the job of delivering the code that belongs to a determined state, and any change in the program language doesn't imply any change in GUIDE's main components. Currently, it was implemented in the WBI, as part of the Javascript API. The advantages of this approach are the full access to the DOM tree and the ability to cope with the dynamically introduced elements as the extraction is made in browser processing time.

The following section details how this extraction is made.

9.3.3 In Browser User Interface Description Language Creation

In order for the UIREC to obtain meaningful information for the several components of the GUIDE framework, some conditions have to be verified in terms of the application's implementation: all the elements that are meant to be adapted and recognized by GUIDE, preferably the majority of the elements presented on the interface, must have a unique id and the respective WAI-ARIA tag describing the role of the element.

The process starts after the application loads its resources (e.g. style-sheets and scripts), and proceeds during the *Framework phase*. Then, the HTML and CSS information is sent to the UIREC, where the parsing takes place in two phases: (1) extraction of the structure of the user interface by parsing the HTML and (2) extraction of the style information of each element by parsing both HTML and CSS information.

The parser goes through the DOM in search for the elements correctly marked with an id and the WAI-ARIA role, and starts forming the structure section of the UIML. The structure is formed with a set of <part>tags with the id and class properties. The id's match with the HTML elements and the classes with the WAI-ARIA roles. There are some roles that encompass other roles, which is the case of *menu* that has *menuitems*. In these cases, the parser takes into account the parent and child elements.

The style section on the UIML corresponds to the properties defined on the style-sheets and/or HTML properties. This section is composed of <property>tags, each one having a *part-name* tag, that matches the *part* id on the structure section, the property *name* (e.g. background-color) and the value. The UIREC can understand specific CSS properties as well as properties defined in CSS classes.

Besides CSS properties, the UIREC needs positioning and size information about the elements. As most of these properties have different representations (e.g. relative positions, percentage, pixels), the WBI has the ability to add to each element GUIDE specific properties. These properties contain *x*, *y*, *width* and *height* absolute pixel values divided by the screen size (e.g. $guidePositionX = 0.124$). The UIREC adds these properties to the style section.

When the processing is finished, the UIML document is sent to the GUIDE framework's bus system and collected by the interested components.

The next section describes what is the real effort made by TV application developers to integrate their applications with the Framework.

9.3.4 Implications for Application Developers

As discussed above, the TV applications considered so far in GUIDE are based on web-based languages like HTML, CSS and JavaScript because of their wide acceptance among developers and general compliance with STB and HybridTV [4] specifications.

In the end, what is expected from the developers is nothing but the specification and integration of WAI-ARIA tags to define the role of the different UI elements and, possibly, some accessibility considerations such as the compliance with WCAG 2.0 guidelines [5]. The only additional effort is the integration with the GUIDE Javascript API, which is accomplished with a very small number of lines of code.

We believe that this effort represents a very small overhead for application developers. A similar or, more probably, an even larger effort, would have to be made if the application was intended to be accessible, in the first place, following current practices.

9.4 Interpreting Input

One of the most crucial aspects of multi-modal interactive systems is interpreting user input, which can either be a simple or complex task, depending on factors such as the number and type of modalities involved, architectural and implementation choices, or even user and contextual requirements.

As a system grows or branches in terms of interaction mechanisms available, so does the amount and variation of information received by it. For this reason, a way of correctly interpreting all of this data is needed, along with adaptation mechanisms that make the interaction experience the most adequate for each user. In addition, users, the way they interact with applications, and their surroundings, can also evolve over time, which forces a fusion engine to constantly adapt its process of decision-making in order to provide trustworthy results.

The GUIDE framework is an user oriented system, capable of providing distinct modalities and devices for providing input, which includes speech and gestures recognition, remote control, among others. For these reasons, a great focus of the framework development was set on creating an approach capable of processing data from these sources, combine it when needed and provide high-level interpretations that are of use to other components of the system.

9.4.1 Requirements for Input Fusion

The main task of the GUIDE fusion engine is to potentially combine any incoming input from recognizers, reach an interpretation of that data and forward it to a dialog manager that continues the process, ending with a response that is returned to the user. The key to provide the most suitable interpretation for a given situation is to take into account critical information from three main sources: input events, a user model and a context model.

When the user is interacting with GUIDE, input recognizers are constantly capturing and generating events that will be sent to the fusion module, which

constitute the base for creating interpretations. These events, which contain temporal (e.g. timestamps) and semantic attributes (e.g. what the user said, which key was pressed) can be considered the most important piece of knowledge, because without it there would not exist a purpose for data fusion. Information about the user, although not essential for understanding input, it is of extreme importance for the fusion process, because it allows to tweak models and algorithms in accordance to each user's necessities and preferences. This type of data is extracted from user models that are constructed a priori by an initialization application, and allow the fusion engine to have access to data such as the level of user proficiency with each modality or device available. Knowing the extent of user capabilities towards the system is an important asset, but it is also quite important to understand how the environment, that surrounds the user, affects the way in which these capabilities are used. The context model is the third component used by the fusion engine to create decisions about what is happening between user and system. The information that must be contained in this model includes, for example, how current environmental conditions are affecting the usage of a certain modality or device (e.g. a noisy room can have a negative impact on speech recognition) or the engagement between user and system (e.g. if the user is passive for a long time, it may need some assistance using an application).

9.4.2 Previous Works on Multimodal Fusion

Multimodal interfaces and ways of interacting with them have been subject of study for the past two decades [6]. This is also true for the process of multimodal fusion, for which there have been envisioned different levels, architectures and algorithms. Sharma et al. [7] considers three levels for fusion of incoming data: sensor-level (or data-level) fusion, feature level-fusion and decision level-fusion. Other authors such as Sanderson and Paliwal [8] define terms with similar meanings such as pre-mapping, midst-mapping and post-mapping fusion. The difference between these types of levels, is essentially, at which time, information combination takes place. Pre-mapping data-level fusion, deals with raw data coming from recognizers, representing the richest form of information possible, quantitatively speaking. Because the signal is directly processed, no information loss occurs, although it is very susceptible to noises and failures. Due to the heavy processing involved, sensor-fusion is most suited for situations where multiple streams of a single modality are involved. Pre-mapping feature-level fusion, is a type of fusion oriented for closely-coupled or time synchronized modalities such as, for example, speech and lips movement recognition. In this type of fusion, features are extracted from data collected by several sensors, and if they are commensurate they can be combined. Unlike data-level fusion, it can suffer from data loss, but manages noise interference better. In midst-mapping fusion several information streams are processed concurrently while the mapping sensor-date/feature space

to decision/opinion space takes place. This type of fusion, similarly to feature-level fusion, is also oriented for closely coupled modalities such as lips and speech recognition.

One of the most common and widely accepted forms of fusion is decision-level fusion, and that is because it allows multi-modal systems to make effective use of loosely-coupled modalities, such as the case of GUIDE. Because the information received by the fusion engine has already been processed, noise and failure are no longer issues to deal with. This means, that fusion will have to rely on preprocessed information in order to construct semantic meaning from combining partial semantic information coming from each input mode. That preprocessed information constitutes a concrete decision that was produced by one or more recognizers. Opinion-level fusion (also called score-level fusion) is very similar to decision-level because both of them operate after the mapping of data/feature-level space into decision/opinion space. In fact, some literature [9] considered the former as a sub-set of the latter. However, in the case of opinion-level fusion, a group of experts provides opinions instead of hard decisions, and for that reason Sanderson and Paliwal [8] found more adequate to make a distinction between the two types. Opinions combination can be achieved, for example, through weighted summation or weighted product approaches, before using a classification criterion (e.g. MAX operator) in order to reach a final decision. The main advantage of using opinion over feature vectors concatenation or decision fusion is that opinions from each expert can be weighted. Fusion classifiers can be distinguished not only by the type of fusion or architecture they possess, but also by whether they are adaptive or non-adaptive [10]. The basic concept around adaptive, or quality fusion, is to assign different weight values associated with a modality. This allows to imprint adaptive features into a system, by setting the reliability and discrimination of experts through time according to the state of the environment, signal quality, users, or application logic.

As for options to implement these ideas and approaches, Dumas et al. [11] considered the following as typical choices for decision-level architectures: frame-based fusion, using data structures called frames or features for meaning representation of data coming from various sources or modalities, modeling objects as attribute-value pairs; unification-based fusion which is based on recursively merging attribute-value structures to obtain a logical whole meaning representation; symbolic/statistical fusion, an evolution of standard symbolic unification-based approaches, which adds statistical processing techniques to the frame-based and unification-based fusion techniques.

Taking into account the low and mid-levels of fusion described, it is clear that these approaches have severe limitations that make them not suitable for the fusion engine of the GUIDE framework, which has to deal with loosely-coupled modalities and a high flow of data that must be handled quickly and efficiently, while at the same time consuming the minimal amount of system resources, which are heavily demanded by other components. As for high-level types of fusion, decision-level is also not a completely optimal solution to embrace, due to the fact that it is not directly oriented for systems that must deal with unpredictability or uncertainty, something that is quite important for GUIDE.

9.4.3 Adaptive Multimodal Fusion in GUIDE

In a system like GUIDE, which runs in a set-top box environment with limited processing capabilities, it is of extreme importance to minimize the workload of each component in the framework. For this reason, the approach taken in designing the fusion module was centered on a high-level type of fusion, namely opinion-level fusion, which assigns a considerable part of the responsibility of understanding input to the recognizers, that provide opinions that must be interpreted and merged into an interpretation. From an architectural point of view a frame-based strategy [11] was chosen due to its simplicity and also because it could easily be augmented to fulfill other requisites such as imprinting adaptability in the system and consider uncertainty in the provided input. This implementation uses data structures entitled *frames*, which consists of two major sets of data. The first one is a set of slots, which can either contain triggers or sub-frames. Triggers are conditions that are to be met in order for the slot to be activated, while a sub-frame is a regular frame contained inside another frame, allowing the representation of more complex interaction scenarios. A trigger is associated with one and only one modality (such as speech or pointing) and contains data related to an input event, that essentially represent user actions. The second set contained inside a frame consists of results, which are interpretations or commands that have to be forwarded to the next component in charge, when frame activation occurs. This activation can occur in different conditions, because, for example, in some contexts we want all the slots conditions to be triggered while in others only one might suffice. Taking into account temporal constraints is also crucial when dealing with input, and for that reason, this frame structure also keeps track of when conditions are met so that dynamic thresholds can be set, in order to allow slower or faster-paced interactions, which is convenient for users that have different characteristics.

The frame-creation process is something that is expected to occur many times during an application life-cycle. As the context of the applications changes, the fusion module must prepare itself to potentially receive different types of input events and send the appropriate responses. To this, whenever the application context is modified, a message is received from the system's dialog manager, containing the representation of the current UI displayed on the screen, expressed in UIML. When the fusion obtains this knowledge the frames creation process can then begin. For each kind of interactive element, specific frames will have to be considered and created. In the case of buttons, for instance, frames will have to be created so that these elements can be clicked using voice or pointing gestures. It also important to notice that the frames created are not solely related to the UI itself. There are other commands that should be available at all times, and are independent of the application context. Examples of such situations would be the user requesting a list of available speech commands or an explanation of a certain element. Upon this process finalization, all the inputs relevant for the current context and their respective responses are defined. As the user produces input, the fusion module assesses at all times which frames can be activated with the input events received.

As previously mentioned, this approach is based on opinions, and therefore it is expected to received, from most input recognizers, a confidence value along with the semantic data involved with the input event. In this way, whenever an event can trigger a certain slot, this slot is also given a confidence value. However, the slot confidence does not depend only on the input, because the fusion module also has to consider the user capabilities and the surrounding environment. For this reason, the fusion engine is constantly being updated, by other components, regarding changes on the user and context, which are also used to calculate the confidence on each slot. For instance, if the system knows the user has difficulties using the speech modality, it will assign a smaller weight to this modality, which in conjunction with a low confidence speech event may trigger a slot. The purpose of defining confidence values for each slot is to attribute an overall confidence value for the frame, which will serve as a mean to compare activated frames and deciding which one is more likely to represent the actions expressed by the user at that point, in order to return the appropriate interpretations.

9.5 Generating Output

Another crucial component of the framework is the multimodal fission module. Fission is responsible for generating the appropriate UI adaptations and for delivering the output to the user using more modalities if necessary, in order to achieve the best possible user experience.

Using a unimodal system limits information presentation into a single modality, this excluding persons who suffer from an impairment to the sensory channel needed to perceive that information (a blind person cannot see graphical information and a deaf person cannot hear sounds). Additionally, a person might be temporarily precluded from using one sensory channel (e.g. her visual attention might need to be focused elsewhere). For these reasons, it can be very important for an application to be relieved of the limitation to present information in a single modality. The multimodal fission component allows applications to present their information using different modalities.

Modes are the sensorial system of a human with which he perceives the world. A modality is defined by the structure of the information that is perceived by the user (e.g. text, sound, vibration, etc.) and a medium is a channel or the mean used to express the modality, i.e., the peripheral devices such as a monitor or a TV screen, loudspeakers and so on. All these three components are dependent on each other's [12]. By combining the information presentation into an adaptive multimedia output, we can enable a more natural and effective interaction whichever mode or combination of modes are best suited to a given situation, context and according to user's preferences and abilities.

The adaptive multimodal fission component is responsible for dividing or selecting the output channels to distribute the information through the available outputs and according to the user profile and environmental context. We also

consider the existence of active and passive outputs, or primary and secondary outputs. For example when using a form of haptic feedback through vibration you can actually hear it too, being vibration the primary output and auditory the secondary output.

9.5.1 Requirements of Multimodal Fission

We defined since the beginning of this project that the developing strategy would be an user centered methodology. This methodology aims to meet the user's requirements, behaviors and specificities by studying and analyzing their interactions with a multimodal system in the most likely end user environment. The target user population considered in the scope of the GUIDE project are elderly people, but many of the findings can be generalized to other population groups.

After conducting user studies and by analyzing the data, we could conclude that applications should always present a short number of interactive elements for each screen, focusing on big buttons. If developers make complex UIs, GUIDE has to be capable of dividing one screen in multiple screens (and provide navigation through them), or present options to the user in alternative modalities. Applications should make sure both text size and audio volume are configurable by the user at the beginning as well as in the middle of an interaction. If the application by itself doesn't offer this option, GUIDE UI adaptation should offer this possibility.

The existence of a strong relation between arm used for pointing and item location on screen, will influence the way developers design the layout of their applications, as it also affects the configuration and parametrization of GUIDE presentation manager, as both have to contemplate the existence of this user-UI relation.

If system feedback and presentation could only be performed in one modality (Avatar, Audio or Visual information), the way to do it would depend on the interaction and application context but also on the user's preferences and capabilities. This is also true for the type of Avatar: head only avatar would be more suitable for talking with the user or giving simple feedback, while half and full-body Avatar would be suitable for instructing the user on how to do certain gestures, or on how to perform certain tasks. However, and independently of which output modality chosen, output should be repeatable every time the user asks for it again, solving problems derived from lack of attention or changes in the context of interaction.

Trials showed that each user has its own characteristics and interaction patterns, but they can be grouped into different clusters. Somehow, information about the user must be collected by the system in order to perform the right adaptation.

To reach the elderly population, characterized by a set of age related disabilities, but traditionally also by some reluctance to adopt new technologies, GUIDE instead of relying on new interaction devices, opted for interaction modalities that are already familiar to the target population. This multimodal interaction scenario might demand a set of skills from its users, not for the interaction itself, but for the

setting up and configuration, that certainly should not be imposed on a regular TV viewer, be him or her elderly or not. As such, in order to achieve the best interaction experience and performance possible from the multimodal interaction set-up, GUIDE includes the possibility to automatically adapt its features and their operation parameters. Another benefit from presenting users with natural interaction modalities, is that they do not require long learning periods in order to be used, thus overcoming another factor that drives away users from new interaction technologies.

A fission component needs to have knowledge of the application's UI and that information must be structured and has to contain all elements and their properties in order to be possible to adapt that content to the user. The Application Environment Interface translates the application state (e.g. a Web page) into UIML to represent the application's UI in a language that is understood by the GUIDE core. The multimodal fission processes the UIML representation of the application's state and decides on how to adapt presentation and interaction aspects. The results of this adaptation are then transmitted to the interface who must translate them for the application. As stated before each user has his own characteristics and set of skills and fission will chose the most appropriate modalities based on an user profile (if two or more users are in front of the TV, their profiles will be merged into a single one). These parameters are divided in two types:

- A set of modalities (input/output) with a value representing their likeability to be used and the level of necessity to be adapted by Multimodal Fission;
- A set of more specific parameters for visual or auditory properties. This data represents the minimum recommendations by the User Model (e.g. the smaller font size a given user can read) and are subject to change by the Multimodal Fission's decision evaluation system.

Adaptations do not depend solely on the user who is interacting with the system. Some contextual elements change how the presentation should be rendered. A good example is the ambient noise in the environment where the system is being used. If the room is full of persons actively chatting with each others, the system will gather the noise ratio level and fission will use modalities alternative to auditory modalities (e.g. avoid auditory messages and use text visual messages). Other example is the user distance to the screen which needs to be considered when adapting visual elements of the screen (e.g. when calculating the font-size or buttons' size). The requested environmental data is the responsibility of a Context Model which fission, and other components, will query whenever needed.

9.5.2 Existing Approaches in Multimodal Systems

Systems that combine outputs evolved since the early nineties where text and graphics were combined (e.g. COMET [13]). More recent systems combine speech, haptic, graphics, text, 2D/3D animations or avatars (e.g. SmartKon [14], MIAMM [15]). Although most applications use few output modalities and consequently straightfor-

ward fission techniques, dealing with the above-mentioned combination of outputs can make the presentations more complex, difficult to coordinate and ensuring coherence.

According to Oviatt [16], fission engines should follow three tasks:

- Message construction. The presentation content to be included must be selected and structured, i.e., it is necessary to decompose the semantic information issued from the dialogue manager into elementary data to be presented to the user. There are two main approaches for content selection and structuring that can be employed – schema-based [17] or plan-based [18]. However, in GUIDE this task begins before the fission component processing begins, since it is constrained by the application's layout and content design.
- Modality selection. After the message construction, the presentation must be allocated, i.e., each elementary data is allocated to a multimodal presentation adapted to the interaction context. This selection process follows a behavioral model that specifies the components (modes, modalities and medium) to be used. The available modalities should be structured according to the type of information they can handle or the perceptual task they permit, the characteristics of the information to present, the user's profile (abilities, skills, impairments and so on) and the resource limitations. Taking this into consideration is necessary for optimal modality selection. For the completion of this goal there are three approaches: rule based [19], composite based [17] and agent based [20].
- Output coordination. Once the presentation is allocated, it needs to be instantiated, which consists in getting the lexical syntactic content and the attributes of the modalities. First the concrete content of the presentation is chosen and then attributes such as modality attributes, spatial and temporal parameters, etc., are fixed. For a coherent and synchronized result of the presentation, all used output channels should be coordinated with each other.

9.5.3 Adaptive Multimodal Fission in GUIDE

GUIDE's fission component, although based on the aforementioned approaches, follows the What-Which-How-Then (WWHT) approach, a conceptual model of Rousseau et al. [12]. This component must know what information to present, which modalities to choose to present that information, how to present it using those modalities and coordinate the flow of the presentation.

In order to select the most appropriate modalities to use, it is necessary to define how deep the adaptation level will be. Three levels of adaptation of the interaction and presentation interface were envisioned, which were characterized as Augmentation, Adjustment and Replacement. These levels represent an increasing change to the visual presentation defined by the application, from no change to the visual rendering to a, possibly, complete overhaul.

Augmentation is the lightest form of adapting the interface implemented by the developer. The visual rendering is not subjected to any change, as GUIDE only complements it with other modalities. Usually, applications are developed using primarily visual presentation mechanisms. As a consequence, audio modalities will, foreseeably, be the most used in such situations in the form of redundant information (e.g. speech synthesis of content presented on screen).

Adjustment is the level of adaptation where the visual interface rendering is adjusted to the abilities of the user and which can also be combined with augmentation. Once again, considering applications are primarily developed taking into account visual rendering, this corresponds to adjusting several parameters of the visual rendering (e.g. font size and contrast). If other modalities are employed, their parameters can also be target of adaptation (e.g. adjusting audio volume).

Replacement level is the most complex adaptation scheme as it means that, not only presentation changes can be made to the application's interface (i.e. the adjustment level), but it can also result in the replacement of some interactive elements for other (e.g. menus for buttons) or even in the distribution of content over different modalities or different screens in case of visual rendering. This level is extremely useful for users with cognitive impairments, who, while navigating through an application, can become lost due to the tangle of menus and buttons displayed. The content of an application state can be simplified and divided by various screens or rendered through other modalities such as the Avatar or speech synthesis.

Given that GUIDE aims to support legacy applications (with changes as small as possible to their code) we must consider that these applications have been developed without accessibility concerns towards impaired users. Ideally, GUIDE would be able to evaluate if the application's presentation is close to the recommended presentation parameters for the current user and context specifications (e.g. the text sizes are between the values perceived by the user's vision), and based on that analysis select which adaptation level to apply. In practice, this represents a loss of control for application developers and publishers which they do not agree to. As such, the level of adaptation an application might be subjected to, can be limited by the application publisher. If the application developer allows, GUIDE can apply the computed adaptations. Alternatively, GUIDE can forward these adaptations to the applications through the GUIDE API, and the application can choose which adaptations to apply, thus retaining a greater degree of control.

The replacement level has been left out of project developments, given two factors: first, the reluctance shown by application developers in having a *foreign* framework taking over the rendering of their applications, and second, the computational requirements for such a complex decision process, might be too much for the processing power of a set-top box. Still, it is envisioned that in the future new algorithms will be devised to this end, replacing current algorithms since the variables at play are substantially different than the ones for the two first adaptation levels.

After the selection of the adaptation level best suited to the situation, the modalities to render the content are chosen through weights selected in accordance

with the availability or resource limitations (context model) and with the user specificities described in the user model. This means that fission uses a rule based system to decide which modalities are the best for a given situation. There are two types of decisions to be made, one is the modalities which will see their content adapted and the other is the modalities that will be used to complement other modalities.

Using the information provided by the user and context models, the fission module is capable of calculating the best values for visual elements within the recommended ones by evaluating the presentation coherency (e.g. assuring that bigger buttons will not overlap each other or reach screen boundaries). Once the presentation is ready to be rendered, the necessary messages to the output devices are sent in a coordinated way. To synchronize the presentation flow, coordination events are sent to the bus in order to start or stop rendering, or to be notified when a render is completed. These rendering instructions are handled by a buffer in the fission module, which sends one instruction for each device at a time. The device will then respond with a notification of completion or failure. By controlling the flow of events sent and notifications received, instructions that do not get a chance to be rendered because a new state needs to be loaded due to user intervention are not even sent to the rendering devices, saving bandwidth and making the whole process more efficient.

9.6 Future Outlook

This chapter presented the GUIDE framework, a solution to endow applications with adaptive multimodal mechanisms, benefiting both end-users, by providing them with natural interaction mechanisms and adapted presentations, and application developers, by empowering their applications with these mechanisms requiring only a small effort.

GUIDE builds on the current trend that is bringing multimodal interaction into the living room. Gaming consoles have started the momentum, but more entertainment appliances with multimodal interaction capabilities have recently entered the market. GUIDE expands on these offerings by integrating more modalities, by complementing multi modality with adaptation capabilities, and by making it easy for platform developers to integrate it into their offerings, be it on the application, platform or interaction device level.

In the coming years, it can be expected that the computing power of set-top boxes and connected TVs will keep increasing, making this solution more viable, powerful and adoptable. The technical restrictions that are still felt will become weaker, and more powerful algorithms will be enabled, thus affording interaction paradigms even more suited to their environments.

Additionally, future users will already be more acquainted with natural interaction modalities, since we are already being more and more exposed to these in current interaction devices, like tablets and smartphones where gesture and speech interaction are becoming common.

Taking all this into consideration, we can expect that this kind of interaction, and its supporting frameworks will enter the marketplace sooner rather than later, and become standard in the living rooms. Furthermore, it is not hard to imagine the next step, where this type of frameworks becomes ubiquitous, supporting personalization and adaptation, not only in the living room, but everywhere where a networked device is available.

References

1. Lochrie, M., & Coulton, P. (2012). Sharing the viewing experience through second screens. In *Proceedings of the 10th European conference on interactive tv and video (EuroiTV '12)* (pp. 199–202). New York: ACM.
2. *Accessible Rich Internet Applications (WAI-ARIA) 1.0*. (2011). From: http://www.w3.org/TR/wai-aria/.
3. Abascal, J., Aizpurua, A., Cearreta, I., Gamecho, B., Garay-Vitoria, N., & Miñón, R. (2011). Automatically generating tailored accessible user interfaces for ubiquitous services. In *The proceedings of the 13th international ACM SIGACCESS conference on computers and accessibility (ASSETS '11)* (pp. 187–194). New York: ACM.
4. *Hybrid Broadcast Broadband TV*. (2012). From: http://www.hbbtv.org/pages/about_hbbtv/introduction.php.
5. Caldwell, B., Cooper, M., Reid, L., & Vanderheiden, G. (2008). Web content accessibility guidelines 2.0 (W3C Note, December 2008). From http://www.w3.org/TR/WCAG20/.
6. Nigay, L., & Coutaz, J. (1993). A design space for multimodal systems: Concurrent processing and data fusion. In *Proceedings of the INTERCHI 93 conference on human factors in computing systems* (pp. 172–178). New York: ACM.
7. Sharma, R., Pavlovic, V. I., & Huang, T. S. (1998). Toward multimodal human-computer interface. *Proceedings of the IEEE, 86*, 853–869.
8. Sanderson, C., & Paliwal, K. K. (2002). *Information fusion and person verification using speech & face information* (Research paper IDIAP-RR 02-33).
9. Hall, D. L., & Llinas, J. (2001). Multisensor data fusion. In D. L. Hall & J. Llinas (Eds.), *Handbook of multisensor data fusion* (pp. 1–10). Boca Raton: CRC Press.
10. Poh, N., Bourlai, T., & Kittler, J. (2010). Multimodal information fusion. In J.-P. Thiran, F. Marqués, & H. Bourlard (Eds.), *Multimodal signal processing theory and applications for human computer interaction* (p. 153). Oxford: Academic Press.
11. Dumas, B., Lalanne, D., & Oviatt, S. (2009). Multimodal interfaces: A survey of principles, models and frameworks. *Human Machine Interaction, 5440*(2), 3–26.
12. Rousseau, C., Bellik, Y., & Vernier, F. (2005). WWHT: un modele conceptuel pour la presentation multimodale d'information. In *IHM* (Volume 264 of ACM international conference proceeding series, pp. 59–66). New York: ACM.
13. Feiner, S. K., & McKeown, K. R. (1993). *Automating the generation of coordinated multimedia explanations* (pp. 117–138). Menlo Park: American Association for Artificial Intelligence.
14. Reithinger, N., Alexandersson, J., Becker, T., Blocher, A., Engel, R., Lockelt, M., Muller, J., Pfieger, N., Poller, P., Streit, M., et al. (2003). SmartKom: Adaptive and flexible multimodal access to multiple applications. In *Proceedings of the 5th international conference on multimodal interfaces (ICMI 2003)* (pp. 101–108). New York: ACM.
15. Reithinger, N., Fedeler, D., Kumar, A., Lauer, C., Pecourt, E., & Romary, L. (2005). Miamm – A multimodal dialogue system using haptics. In J. van Kuppevelt, L. Dybkjaer, & N. O. Bernsen (Eds.), *Advances in natural multimodal dialogue systems*. Dordrecht: Springer.
16. Oviatt, S. (2003). Multimodal interfaces. In A. Sears & J. Jacko (Eds.), *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging application* (pp. 286–304). Hillsdale: L. Erlbaum Associates Inc.

17. Fasciano, M., & Guy, L. (2000). Intentions in the coordinated generation of graphics and text from tabular data. *Knowledge and Information Systems, 2*, 310–339.
18. Duarte, C. (2008). *Design and evaluation of adaptive multimodal systems*. Phd thesis, University of Lisbon.
19. Bateman, J., Kleinz, J., Kamps, T., & Reichenberger, K. (2001). Towards constructive text, diagram, and layout generation for information presentation. *Computational Linguistics, 27*, 409–449.
20. Han, Y., & Zukerman, I. (1997, March). A mechanism for multimodal presentation planning based on agent cooperation and negotiation. *Human-Computer Interaction, 12*, 187–226.