Toolkits for Multimodal Interaction on Mobile Devices [*]

Jan Aurel Kühni
University of Fribourg
Department of Informatics, DIVA Group
1700 Fribourg, Switzerland
jan.kuehni@unifr.ch

ABSTRACT

With the appearance of mobile devices that provide a wide range of input modalities, application development that incorporates and makes use of these modalities has become an increasingly difficult and time consuming task. In recent years, several frameworks and toolkits have emerged that encapsulate the reoccurring, low level implementation logic of multimodal applications. Many of these approaches differ from each another and emphasize on different aspects of the multimodal interaction. This paper discusses some recent framework approaches and elaborates their specific advantages and drawbacks for developers.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: [Design Tools and Techniques - User Interfaces]; H.5.1 [**Information Interfaces and Presentation**]: [Multimedia Information Systems]

General Terms

Multimodal Interaction, Mobile Devices, Framework, Component, Fusion, Evaluation

1. INTRODUCTION

While mobile devices have strongly increased in computational power over the last years, the ability for the user to interact with such types of devices is disturbed by the small screen size and the lack of a physical, touch responsive keyboard. Alternative ways of interacting have emerged, driven by the development of advanced input modalities such as touch gestures, voice interaction, gaze tracking and the use of accelerometers or gyroscopes. These new input modalities constitute a more natural way of interacting with a device and eliminate the previously established interaction boundary found in simple mouse and keyboard control.

While many of todays applications on mobile devices utilize these new modalities to a certain extent, the true potential of multimodal interaction arises through the combination of different interaction methods. Unfortunately, the prototyping, implementation and testing process of such applications remains a difficult task. In recent years, a series of frameworks and toolkits appeared, that allow developers to more easily conceive and implement multimodal applications on mobile devices. The need for such frameworks arises by the fact that a large part of the implementation process has to be repeated every time a new multimodal system is developed. This reoccurring, complex implementation logic can be encapsulated within a framework and reused in a different system. While many of the multimodal frameworks focus on hiding underlying logic, they vary from each other in the way they emphasize on different aspects. We will try to elaborate these differences and discuss their respected advantages and drawbacks.

[*]The Paper is a deliverable of the Msc Research Seminar Multimodal Interaction on Mobiles Devices in 2012, DIVA Group University of Fribourg, supervised by Dr. D. Lalanne

2. MULTIMODAL FRAMEWORKS

In its core, a framework is a "reusable, semi-complete application that can be specialized to produce custom applications" [4]. In terms of multimodal interaction, a framework must allow a developer to more easily create applications that use and combine a set of multimodal interaction techniques. This specification can be achieved by identifying which elements of a multimodal system can be abstracted away from the application itself and encapsulated within a framework. A well designed toolkit will hide most of the internal complexity and provide all the needed 'how' functionality, the user then supplies the 'what' knowledge, usually by implementing abstract framework functions.

In general, depending on the type of application and its desired functionalities, the implementation process for creating multimodal systems can vary greatly. Yet, global requirements for such applications remain unchanged: at its core, every multimodal system requires a way to interface, communicate and fetch new data from all used input devices and sensors. In the context of mobile devices these include, but are not limited to, touch screens, accelerometers, microphones, cameras and possibly other external components. With new devices and input modalities appearing at a regular basis, interfacing sensors and input devices is a central aspect in the development of multimodal applications. In order for the toolkit to be generally applicable, it needs to provide some functionality to describe and interface currently unknown modalities in an abstract manner.

Raw input data must then be processed, transformed and possibly recombined into a new data set, which is then made available to the system. We will see that toolkits focusing on advanced fusion techniques offer a very powerful way to combine input modalities through the use of probabilistic methods. The altered input data can now be interpreted by the application in order to generate a decision and issue a response through the devices actuators.

The following frameworks and toolkits will all tackle these core implementation aspects, but choose different ways to achieve a viable solution. We will first discuss component based approaches, in which an iterative design process is used to abstract all parts of the multimodal system into clearly defined components. A second approach then focuses on fusion techniques that allow for advanced coupling of input modalities. Further we will discuss a markup language approach, aiming to create a new standard for describing multimodal interaction. Finally, we will look at web based approaches that limit the amount of supported modalities in order to be deployable across multiple device families.

2.1 Modularization

Frameworks focusing on modularization try to abstract all the different aspects of a multimodal interaction system into well-defined elements that can later be interlinked with each other. The earlier multimodal frameworks such as MATCH [7] were based on finite state architecture, where individual agents (responsible for certain tasks) communicated together through a central node element. Since the architecture dates back to 2002 and is not well suited for reuse in different applications, it will not be discussed in detail.

The ACICARE [9] framework implements this modularization by introducing two distinct ICARE components. Each aspect of the input transition, from the input devices to the combination of modalities, can be achieved by interconnecting clearly defined modules. The framework uses an iterative design approach and can be used for rapid prototyping and development of multimodal applications. It extends the ICARE framework which was originally developed for desktop computers by adding support for mobile devices. Elementary components define pure interaction modalities. This functionality is achieved by connecting a physical interaction device that acquires information with an interaction language. Through the use of a specific grammar, the interaction language defines a set of meaningful expressions, which can be applied to the sensor input. The grammar describes how the application reads and interprets input from the input device components. Device components include the keyboard, microphone and dedicated key devices - language components depend on the application itself. The elementary components make up the building blocks for defining and describing input modalities. To combine these modalities, ACICARE introduces the idea of generic composition components. These components are used to specify the combined usage of modalities. They are based on the CARE properties complementary, redundancy, equivalence and redundancy/ equivalence and allow the developer to define in which way the application state can be modified. The reusability of generic components in ACICARE is rather low - for every new input modality, a new grammar must be written that handles incoming input from the specified modality.

The openInterface [10] framework circumvents the need for a specific grammar by introducing the notion of generic and tailored components. Similar to ACICARE, openInterface is a component-based tool for rapidly developing multimodal input interfaces. Previously, component based frameworks were limited due to low abstraction levels on the implemented modules. OpenInterface increases abstraction by focusing on a component model that includes a wide range of predefined, generic elements. These components implement highly reusable functionalities by defining generic input operations on a higher abstraction level.

The openInterface model incorporates three main component classes.*Devices* describes objects that act as physical input devices used for interacting with the application. The device components are then connected to a transformation chain which is made up of transition and composition components. Generic transformation components act on the input and include command filters (generate commands from a mapping table), threshold filters (filter on input values specified by threshold) and continuous-discrete components (modify input frequency). Some generic transformation components can have a large impact on how device input is interpreted within the application. Continuous zooming could for example be translated into step by step zooming simply by adding a Continuous-Discrete Component to the transformation chain. Composition components can be used to recombine input modalities according to the selected components property. When all transition components are traversed, input data is assigned to tasks that interact with the functional core of the application implementing the openInterface.

Compared to ACICARE, the openInterface has the advantage to offer a graphical user interface called OIDE. The environment allows for simple drag and drop functionality in order to create/manipulate device components and the transformation chain. Within the interaction pipeline, individual components can be switched on and off by a simple button click. This approach allows for very quick prototyping and testing of a multimodal application design.

2.2 Fusion Engines

When dealing with multimodal interaction, advanced fusion engine have become increasingly popular in recent years. Fusion goes beyond a simple modality combination; it describes the process of merging parallel input modalities, aiming to extract additional information about otherwise hidden context information. The fusion process can be implemented in a number of ways. Most commonly, frameworks use an approach based on probability scores. As a fusion example, the characteristics of vocal input (pitch, keywords e.g.) could change the way an application interprets gestural touch input. Though the concept of fusion is in itself easy to grasp, implementation can be a tedious and time consuming task. RDMI [4] is a recent, fusion based framework proposed by Dutch researchers, that uses a new fusion approach and aims to be reusable across applications and modalities. The main focus lies on how spoken commands can be altered or recombined through the use of other modalities. A fusion and a dialog manager make up the core aspects of this object-oriented framework. The fusion process starts with a natural language parser that generates a semantic parse tree from spoken input. Through the use of resolving agents and the fusion engine, the parse tree is then transformed into a set of command frames. This process is achieved by mapping natural language concepts from the input command to application concepts. Based on probability scores, the fusion manager picks a slot element from the previously written frames and forwards it to the dialog manager. Only the dialog manager can make direct calls to the application which implements the fusion framework.

Prior to the actual fusion process, multimodal input has to be acquired from the input modalities. RDMI uses context providers to abstract input devices and to create a uniform access point for resolution agents. The context providers exist as objects which furnish access to some external, possibly abstract data source. The agents act as links between the context providers and the fusion engine, and aim at resolving input ambiguities and mapping input to application concepts. The agents perform mapping by attributing probability scores to their fragment of the initial parse tree and then write the results into command frames. Every agent has only access to one single modality and does not perform fusion itself. Modality fusion happens when the fusion manager picks frames based on their probability scores and forwards them to the dialog manager. This design approach keeps the agents simple and reusable.

Another fusion oriented framework is proposed by Gregory Biegel and Vinny Cahill [1]. It focuses specifically on the development of context-aware applications on mobile devices. The implementation is based on a series of loosely coupled sentinent objects, which represent an abstraction for sensors/actuators and are placed and interconnected within a fusion network. A sentiment object acts as an abstraction to a sensor or higher level context and perceives the state of the environment. In terms of the event-based communications model, a sentinent object consumes various sensor data events as they occur and reacts to the stimulus by generating a response event that is propagated through the network. These individual sub contexts can then themselves be interconnected to form a higher level hierarchy. In order to generate responses, the sensorial input data travels through a fusion network, where the actual modality fusion takes place within the inference engine of each sentinent objects. Compared to RDMI, modality fusion occurs at every step of the sentiment hierarchy and not only at the top. The inference engine is based on probabilistic sensor fusion that is implemented on the basis of Bayesian network theory. Bayesian networks allow reasoning about conditional probabilities and to select the most probable outcome given a certain input. As the multimodal input data travels through the fusion network, a decision generated by the top level sentiment object. The decision is then used to make an application calls and to generate a response through the actuators of the mobile device.

As previously seen in ACICARE, this fusion framework also provides a visual tool that allows for a graphical construction of multimodal applications. The developer simply specifies needed sensors and actuators and interconnects these objects within a fusion network by specifying hierarchies and conditional rules. Even though graphical user interfaces are less common in fusion oriented frameworks, they are in general well suited in the context of multimodal application design. The processing chain from the input modalities to the application call is highly modular and can be visually represented.

2.3 Standardization and Representation Language

While the previously discussed frameworks work well for their specifically designed purpose, they are written in a specific programming language and are thus restricted to a certain device type. Another common problem these frameworks share is the fact that communication among input components is generally not handled in a homogenous manner and adding new input devices might not always be possible. In 2009 there has been an effort to create a representation language that allows for a standardized description of multimodal applications. The resulting EMMA (Extensible Multimodal Annotation Markup Language) [10] framework has been set up as a W3C standard that uses XML markup language to describe all required stages of capturing and processing different input modalities. In this regard, EMMA is less a framework than a generalized representation language.

A typical EMMA XML document contains a series of container elements that, in combination with a set of standardized annotations, describes the used medium together with its modalities. The annotation attributes are used to provide a various amount of metadata, describing the input modalities (timestamps, confidence scores e.g.). In addition, interpretation elements within the markup language act as containers to semantically represent multimodal user input. In order to allow for input transformation and combination, EMMA provides container elements that can be used to group inputs together and to specify an input sequence. The markup language also touches fusion concepts by providing container elements that describe overlapping input modality interpretation by using confidence intervals and other language structures. This aspect is important for multimodal systems using spoken word or gestures as input, where there may be multiple possible interpretations for user intentions. To make it clear, EMMA is not a standalone framework and does not standardize the semantic representation of inputs, it much rather provides standardized containers for application specific markup and a series of standardized annotations to describe commonly used metadata. In order to perform fusion, EMMA must be hooked up to a fusion server that parses the previously created XML documents. The WAMI (Web Accessible Multimodal Interface) toolkit also aims at providing platform independent support for creating standardized multimodal applications. Contrary to the previously discussed native applications, the WAMI toolkit is entirely web based and uses wide spread web standards, such as AJAX and XML. It is not specifically aimed at mobile devices, but as the provided sample projects show, multimodal applications are accessible through any device disposing of a standard web browser. WAMI extends the capabilities of a previously release web accessible framework called WebGALAXY [8]. The framework is built around the *core platform* that implements a client-server model. The components running on the server provide services such as speech recognition, synthesis functionalities and logger components. The client renders an application specific graphical user interface and exchanges information with the server through an AJAX powered controller. The downside of WAMI lies in its focus on speech based interaction, additional modalities such as the camera or accelerometers are not yet integrated within the framework.

3. DISCUSSION

Before choosing a framework to create a multimodal application, developers should precisely evaluate which modalities will be utilized and how they will be processed by their system. Depending on their specification for sensor access, input transformation/combination and application interaction, a toolkit should be chosen which best accentuates the needed aspects.

Frameworks focusing on modularization provide simple drag and drop modules for designing and implementing multimodal applications. OpenInterface supplies a wide range of predefined containers that can be used without further modifications. Generic containers within the toolkit provide abstract properties to transform multimodal input. If more complex, user tailored components are needed, ACICARE provides an interaction grammar hat can be used to write and define user specific modules. The two framework designs result in a tradeoff between simplicity of use and the possibility to implement more complex input manipulation, not foreseen by the framework engineers. While frameworks focusing on modularization are easy to use and well suited for certain types of applications, their capacity to combine and merge different modalities is limited by the provided compositional components and interaction language. When working with a large range of input modalities from which higher level information should be derived, advanced fusion engines offer a more powerful way for developers to synthesize input modalities. The downside of the fusion approach lies in its increased complexity and need for developers to understand and create fusion networks themselves. It might seem that fusion based approaches are superior to component oriented frameworks, because their way of combining and extracting higher level information is more sophisticated. While this is true, they are often limited in terms of individual input transformation.

If the application development focuses on making the product available to a wide range of devices and platforms, frameworks such as EMMA and WAMI provide a more standardized solution. The WAMI toolkit can easily be deployed on the web but is strongly limited in terms of the modalities it supports. EMMA incorporates a platform independent standard and can handle a wide range of input modalities. Unlike other frameworks it depends on external components which have to be provided by the user.

4. CONCLUSION

There exist a wide range of multimodal framework designs aimed at supporting mobile devices. While all frameworks aim at encapsulating low level complexity and allow for faster application development, they focus on different aspects of the information processing chain that occurs between the sensor input, its transformation/combination of sensor data and the way it is used and interpreted to make application calls.

A general problem which is commonly seen in most of the discussed frameworks is the lack of platform support and the missing possibility to make the framework available to a broader audience. Most discussed frameworks run well in clearly defined lab environments, a setup which is not easily reproductible by novice developers. Framework should greatly reduce development time of multimodal applications and therefore be usable in an out of the box manner and support a multitude of platforms and devices. Future research needs to thrive for software solutions that standardize the communication of various input modalities over a wide range of platforms while providing powerful ways to transform and fuse multimodal input.

References

[1] G. Biegel and V. Cahill. A framework for developing mobile, context-aware applications. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, PERCOM '04, pages 361–, Washington, DC, USA, 2004. IEEE Computer Society.

[2] J. H. Dirk Buehler, Wolfgang Minker and S. Krueger. Flexible multimodal human-machine interaction in mobile environments. In *7th International Conference on Spoken Language Processing*, ICSLP-2002, pages 169–172, Denver, CO, USA, 2002. ACM.

[3] S. Eldwaib and R. Pooley. Reusable framework for location-aware mobile applications. In *Proceedings of the 6th International Conference on Mobile Technology, Application &#38; Systems*, Mobility '09, pages 22:1–22:4, New York, NY, USA, 2009. ACM.

[4] F. Flippo, A. Krebs, and I. Marsic. A framework for rapid development of multimodal interfaces. In *Proceedings of the 5th international conference on Multimodal interfaces*, ICMI '03, pages 109–116, New York, NY, USA, 2003. ACM.

[5] A. Gruenstein, I. McGraw, and I. Badr. The wami toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces. In *Proceedings of the 10th international conference on Multimodal interfaces*, ICMI '08, pages 141–148, New York, NY, USA, 2008. ACM.

[6] M. Johnston. Building multimodal applications with emma. In *Proceedings of the 2009 international conference on Multimodal interfaces*, ICMI-MLMI '09, pages 47–54, New York, NY, USA, 2009. ACM.

[7] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. Match: an architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 376–383, Stroudsburg, PA, USA, 2002.

[8] G. F. R. Lau and V. Zue. Webgalaxy: beyond point and click - a conversational interface to a browser. In *Computer Networks and and ISDN Systems*, pages 1385–1393, 1997.

[9] M. Serrano, L. Nigay, R. Demumieux, J. Descos, and P. Losquin. Multimodal interaction on mobile phones: development and evaluation using acicare. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, MobileHCI '06, pages 129–136, New York, NY, USA, 2006. ACM.

[10] M. Serrano, L. Nigay, J.-Y. L. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef. The openinterface framework: a tool for multimodal interaction. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI EA '08, pages 3501–3506, New York, NY, USA, 2008. ACM.