M3I: A Framework for Mobile Multimodal Interaction

Andreas Möller [1], Stefan Diewald [1], Luis Roalter [1], Matthias Kranz [2]

Distributed Multimodal Information Processing Group, Technische Universität München[1]
Lehrstuhl für Informatik mit Schwerpunkt Eingebettete Systeme, Universität Passau[2]

Abstract

We present *M3I*, an extensive multimodal interaction framework for mobile devices, which simplifies and accelerates the creation of multimodal applications for prototyping and research. It provides an abstraction of information representations in different communication channels, unifies access to implicit and explicit information, and wires together the logic behind context-sensitive modality switches by a rule-based approach. In this paper, we present the structure and major features of our framework, and show exemplary implementations of interaction modalities with help of *M3I*.

1 Introduction and Motivation

Multimodality denotes the usage of more than one modality (simultaneously or sequentially) for input and output (Oviatt 1999). Particularly in mobile settings, individual unimodal interaction modes can suffer from limitations (e.g., the *small screen* problem when only the visual channel is used). Multimodality is considered to be more efficient (Oviatt 1999) and more natural (Jokinen & Hurtig 2006). While mobile multimodal systems have been investigated in research (Möller et al. 2012a, Möller et al. 2012b, Schuster et al. 2011), the adoption of multimodality in the real world is still cautious, both from user and programmer side. With the *M3I* (*M*obile *M*ulti*M*odal *I*nteraction) framework, we aim to foster the development of multimodal applications and the system-wide use of multimodal interaction on mobile devices. We respect two notions of multimodality: context-based selection of output modalities, and novel multimodal input methods. To the best of our knowledge, *M3I* is the first approach that satisfies all of the following requirements:

- Holistic support of modeling, prototyping and implementation of multimodal behavior in mobile systems (often, only individual stages, such as prototyping, are supported)
- Support of multimodal input *and* output (other approaches focus either on the interpretation and fusion of input events, or on output modality switching by tasking applications)

- Support of context-driven multimodality (existing approaches are either pure context toolkits, or focus on multimodal interaction without integrating context, at least not using context for both in- and output)
- Applicability in standard development environments, without compiler adaptations (Schuster et al. 2011), and with state-of-the-art mobile operating systems (earlier approaches are dedicated to legacy or desktop platforms providing limited context support)
- Autonomous operation, without server-based evaluation or control through the cloud as e.g. in *Code in the Air* (Ravindranath et al. 2012)
- Definition of multimodal behavior based on *rules,* which is close to the human understanding of automated switching (approaches that use automata, state machines, etc., might comply less with the human mental model of such systems)

2 Mobile Multimodal Interaction Framework

M3I is implemented in Java as Android library. Figure 1 shows, on a conceptual level, the structure of the framework and its basic components. The key elements for realizing rule-based evaluation are `Statements` and `LogicalExpressions`, which describe explicit and implicit events, and `Triggers`, which describe a behavior of the system. Examples for statements are "the user is at home" or "the battery is charged more than 50%". They consist

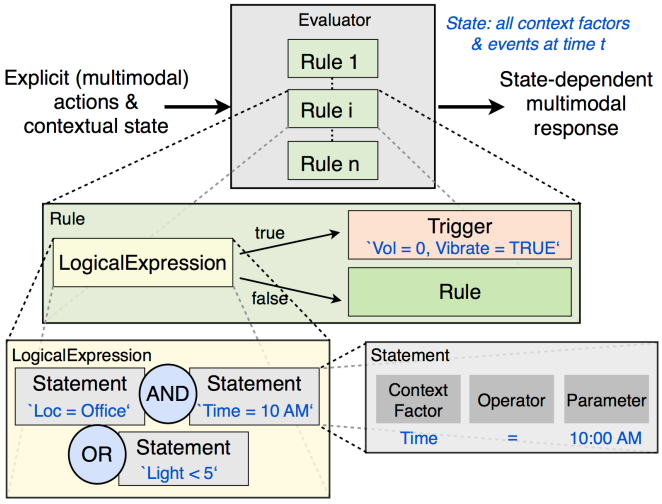

Figure 1: General Structure of the M3I *Framework. Rules define the system's behavior in response to explicit user actions and/or to implicit context factors and events. Thereby* M3I *realizes context-dependent modality changes or novel interaction methods involving different modalities.*

of a `ContextFactor` referring to the subject of the statement, and of an `Operator`, which allows numeric comparisons, within-range-tests, or regular expressions. `Rules` associate the events described by a `LogicalExpression` with appropriate `Triggers` that define what happens if the rule is true or false. Statements can be combined with, e.g., AND, OR, XOR, and NOT to logical expressions, and by recursively nested rules. The set of active rules is evaluated in the framework's `Evaluator` in regular intervals, which are variable to account for time-critical sensor data as well as for battery-conserving location updates.

The rule-based wiring and evaluation mechanism allows defining multimodal in- and output with consideration of context information. Under context, we subsume implicit information, such as the user's current location or activity that impact the choice of output modalities, but also explicit user actions like (e.g., accelerometer-detected) motion gestures, which are themselves part of an interaction. *M3I* formalizes such information as `ContextFactors` and supplies their current value by context methods. *M3I* economizes and simplifies accessing contextual information for the programmer, as it saves the overhead for initializing system services, for creating event listeners etc., and makes all functionality accessible by homogenous interfaces. Basic activity recognition and classification routines abstracting from pure sensor readings are included. *M3I* currently integrates over 50 context factors regarding, e.g., location, ambient noise and light level, device orientation, or proximity information.

Novel interaction methods that incorporate multiple modalities can be realized by linking data from different `Statements` using `LogicalExpressions`. Like this, explicit interactions, such as physical button presses or touch interactions can be intercepted and combined with implicit contextual information. *M3I* also unifies synchronous information and asynchronous information (making underlying *push* or *pull* paradigms transparent). By combination of validity settings and event timestamps, both parallel and sequential actions (e.g. for multi-step motion gestures) can be evaluated.

On output side, `Triggers` realize modality switches and thereby abstract how information should be represented. One notification can be provided, e.g., as visual, haptic or auditory response. States or contexts can be announced system-wide through an Android content provider. That way, each app can provide its own adaptations to state changes, accounting for the fact that interaction modes are often task-specific. An application can, e.g., offer a user interface with larger touch controls that are easier to hit when the user is on a walk.

M3I can easily be extended by feeding results of any method or callback into the framework's decision logic. Triggers are likewise not limited to predefined modalities or actions: a `MethodTrigger` allows implementing custom functionality by calling self-implemented methods. Unlike other frameworks, *M3I* is not focused on or built around an individual use case, but from scratch designed for a broad range of application areas.

3 Exemplary Implementations of Modalities

To prove *M3I*'s suitability for rapid prototyping, we implemented three exemplary multimodal interaction methods (see Figure 2). With *Raise&Call,* the user can bring the phone to

the ear to start a call, combining a device motion gesture and speech as input modalities. With *Press&Shoot*, the camera app is started by pressing the volume button in upright position (motion gesture and button press), and *Pinch@Home* launches the maps application by mimicking a pinch-to-zoom gesture on the home screen (motion gesture, touchscreen). These three input methods were implemented with *M3I* in straightforward manner by a few simple rules (e.g., involving the accelerometer pose as implicit sensor reading and the button press as explicit action for the *Press&Shoot* method). *M3I*'s built-in abstractions (predefined device poses like "upright", "lying on the table", "display up/down") additionally accelerated the development of the *Raise&Call* and *Press&Shoot* methods.

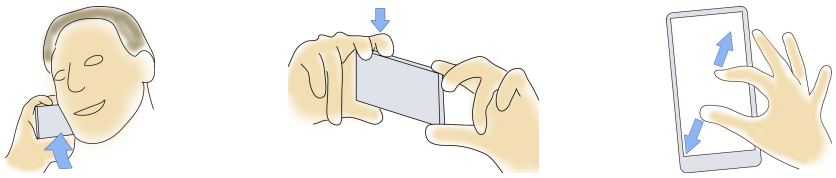

Figure 2: Exemplary interaction methods implemented with M3I*:* Raise&Call*,* Press&Shoot*, and* Pinch@Home

Future work will investigate automated deduction of desired modalities. When a user e.g. mutes the device every week at a certain location and time (probably because of a weekly meeting), *M3I* could proactively suggest this action. Unlike classical supervised learning where the learned model is not visible to the user, the rules to be created could be edited and reviewed to establish transparency about an application's multimodal behavior.

References

Jokinen, K., and Hurtig, T. (2006). User expectations and real experience on a multimodal interactive system. *Proc. INTERSPEECH.*

Möller, A., Diewald, S., Roalter, L., and Kranz, M. (2012). MobiMed: comparing object identification techniques on smartphones. *Proc. NordiCHI,* 31-40, ACM.

Möller, A., Kray, C., Roalter, L., Diewald, S., Huitl, R., and Kranz, M. (2012). Tool support for prototyping interfaces for vision-based indoor navigation. *Proc MobiVis at Mobile HCI.*

Oviatt, S. (1999). Ten myths of multimodal interaction. *Communications of the ACM,* 42(11), 74-81.

Ravindranath, L., Thiagarajan, A., Balakrishnan, H., and Madden, S. (2012). Code in the air: simplifying sensing and coordination tasks on smartphones. *Proc. HotMobile '12*, ACM.

Schuster, C., Appeltauer, M., and Hirschfeld, R. (2011). Context-oriented programming for mobile devices: JCOP on Android. *Proc. COP '11*, ACM.

Contact Information

{andreas.moeller, stefan.diewald, roalter}@tum.de, matthias.kranz@uni-passau.de