Engineering affective computing: a unifying software architecture

Alexis Clay
ESTIA, LaBRI, Université Bordeaux 1 CNRS
Technopole Izarbel, 64210 Bidart, France
a.clay@estia.fr

Nadine Couture
ESTIA, LaBRI, Université Bordeaux 1 CNRS
Technopole Izarbel, 64210 Bidart, France
n.couture@estia.fr

Laurence Nigay
IMAG-LIG
B.P. 53, 38041 Grenoble cedex 9, France , 385, rue de la Bibliothèque, Domaine Universitaire,
laurence.nigay@imag.fr

Abstract

In the field of affective computing, one of the most exciting motivations is to enable a computer to sense users' emotions. To achieve this goal an interactive application has to incorporate emotional sensitivity. Following an engineering approach, the key point is then to define a unifying software architecture that allows any interactive system to become emotionally sensitive. Most research focus on identifying and validating interpretation systems and/or emotional characteristics from different modalities. However, there is little focus on modeling generic software architecture for emotion recognition. Therefore, we propose an integrative approach and define such a generic software architecture based on the grounding theory of multimodality. We state that emotion recognition should be multimodal and serve as a tool for interaction. As such, we use results on multimodality in interactive applications to propose the emotion branch, a component-based architecture model for emotion recognition systems that integrates itself within general models for interactive systems. The emotion branch unifies existing emotion recognition applications architectures following the usual three-level schema: capturing signals from sensors, extracting and analyzing emotionally-relevant characteristics from the obtained data and interpreting these characteristics into an emotion. We illustrate the feasibility and the advantages of the emotion branch with a test case that we developed for gesture-based emotion recognition.

1. Introduction

Many interactive systems [8, 5] that have been developed are based on recognition of emotions. However they have been developed in an ad-hoc way, specific to a kind of recognition model or a particular system. Some existing tools attempt to be more generic such as the EyesWeb application [4] for emotion recognition.

In this paper we adopt a unifying approach by providing a generic software architecture for emotion recognition. Our software architecture relies on a data flow network from raw data captured from sensors to a recognized emotion that can then be exploited within the interactive system. The originality of our approach is to rely on results from multimodal human-computer interaction and their canonical reference architecture models.

The structure of this paper is as follows: we first give an overview of the overall architecture for interactive applications with a specific branch for emotion recognition. Finally we explain how we implement them by adopting a component-based approach and illustrate our approach by presenting emotion software based on the recognition of the emotion conveyed by the observed subject.

2. Overall architecture

2.1. A three level process

Computer-based emotion recognition typically relies on a three step process. Those steps match with abstraction levels usually named signal, feature and decision levels. In this paper, we refer to these levels as capture, analysis and interpretation levels. Capture level regroups the sensors' software interfaces that allow acquiring information about the real world and especially the user. The obtained data is usually at a low level of abstraction but might be produced by complex processing (e.g. In the case of a camera-based full-body tracking system): in this case there are several Capture-level representational systems. Within the analysis level, emotionally-relevant cues are extracted from the captured data. Cues can cover several layers of abstraction and rely on each other: for example in Infomus Lab's work on expressive gestures [3], quantity of motion is used as an emotional characteristic but also as a tool to segment motion into pauses and gestures and, ultimately, computing the directness of a gesture. This example illustrates the case of a sequence of Analysis-level representational systems. Interpretation level is dedicated to the interpretation of those cues to obtain emotions. Several redundant or complementary interpretations can be performed at the same time in order to increase accuracy. Interpretation of a set of emotionally-relevant features depends on several factors. The main one is the choice of emotion theory that was made when designing the emotion recognition software. The choice of a discrete model or a continuous or componential one greatly shapes how the interpretation is performed and how the recognized emotion is communicated to the rest of the system. The set of emotions that are recognizable by the system has a similar impact.

In this work, we lay down three limitations when considering computer-based affective states recognition. Firstly, following the taxonomy of [13], we only consider emotions, due to their temporal aspect: emotions are quick and highly synchronized responses to stimuli. Secondly, we only consider passive recognition, i.e. when the user doesn't thoughtfully initiate a communication to notify the system of his emotional state. We only consider systems where sensors passively monitor the user and the real world. Thirdly, we do not consider systems that learn from a particular user, thus being able to model his personality to better infer an emotion.

2.2. The emotion recognition branch

The emotion branch can be integrated within canonical software architecture models for interactive systems. In Figure 1, we consider two key software architecture models for interactive systems, namely the ARCH reference model [14] and the agent-based MVC model [9]. For adding the emotion branch within the ARCH model, we apply its branching mechanism as shown in Figure 1.a. For the case of the MVC agent of Figure 1.b, we consider a new facet (i.e., the branch emotion) made of three computational elements.

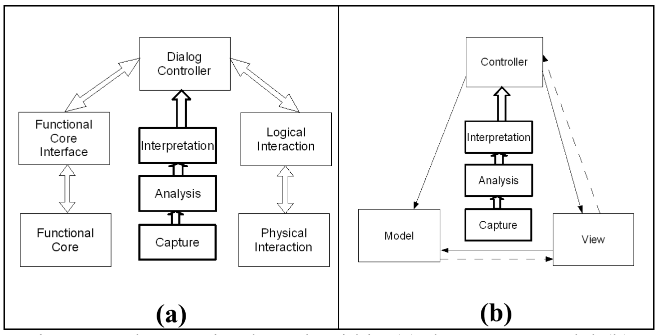

Figure 1. The emotion branch within (a) the ARCH model (b) the MVC Model.

In Figure 1, the emotion branch is connected to the Dialog Controller of ARCH or to the Controller facet of an MVC agent. This is, however, not always the case. We identified three cases that correspond to different roles that emotion can play in an interactive system.

Case 1: As shown in Figure 1, users' emotion can have a direct impact on the Dialog Controller (DC). The DC has the responsibility for task-level sequencing. Each task or goal of the user corresponds to a thread of dialogue. In this case where the emotion branch is connected to the Dialog Controller, the tasks and their sequence can be modified according to the recognized emotion. For example, in an interactive training system, recognition of sadness or anger of the user (i.e., the learner) could trigger the appearance of a help dialog box about the current exercise. Moreover in the driving simulator [1] as well as in the Multimodal Affective Driver Interfaces [10], alarms are presented according to the current recognized state of the driver, modifying the task-level sequencing and therefore the Dialog Controller.

Case 2: The recognized emotion can be manipulated by the Functional Core branch (i.e., Functional Core Interface and Functional Core components of ARCH) as shown in Figure 2.a. The recognized emotion is therefore a domain object. This is the case in the augmented ballet dance show [16] where the recognized emotion conveyed by the dancer is presented to the audience.

Case 3: The detected emotion can have an impact over the Interaction branch as shown in Figure 2.b. For example, a recognized emotion might trigger the change of output modalities (e.g., reducing the frustration of the user). For input interaction, emotion detection could for example imply a dynamic change of the parameters of the speech recognition engine, making it more robust.

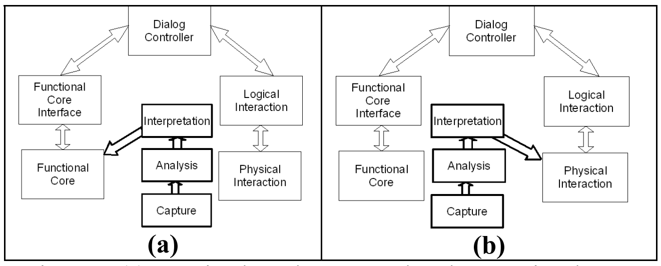

Figure 2. (a) Emotion branch connected to the Functional Core branch. (b) Emotion branch connected to the Interaction branch.

3. Implementation: component-based approach

As for [4], we advocate a component-based model for emotion recognition, in order to ensure modifiability and reusability. A component is a communicative black box; an enclosed piece of processing software that may take and deliver parameters, and subscribe to and deliver data flows. As such, from a system point of view, a component is only known through its interface (in the object oriented programming sense of the word). Our system is hence composed of five component types. Three of them are related to each of the capture, analysis and interpretation level: the capture unit is an interface with a physical sensor, the feature extractor analyses a data flow to extract an emotionally feature, and the interpreter analyses values of a set of cues to deliver an emotion. The two other component types are system-related: adaptors transform data flows format for better modifiability and reusability, and concentrators merge together flows of the same type for increasing robustness. Components communicate with each other using data flows. They can subscribe to one or several data flows as an input, and deliver one or several data flows as an output.

3.1. Underlying concepts: a pure or combined modality

Roughly, a modality is a way of performing a task using a device and an appropriate language to communicate with the machine. Multimodality is the possibility or necessity to use several devices or languages in order to accomplish a task, as illustrated by the "put that there" paradigm [2]. We consider the definition of a modality given in [11]:

$$modality=<d,rs>|<modality,rs>$$

where d is a physical device, and rs a representational system. A representational system is a structured set of signs that is used to communicate with the machine. Interestingly, this definition characterizes the interaction between the user and the system at the physical level and the logical level. This definition is recursive. The recursivity illustrates the fact that there can be a transfer from a representational system to another. For an input modality example, Firefox web browser hosts a plug-in that allows accomplishing tasks (e.g. "go back one page") using mouse gestures (e.g. draw a stroke from right to left). This case is an example of transferred input modality: the modality used is *<<mouse,(x,y)position>, mousegesture>.*

A system is multimodal when several modalities are used to accomplish a task. Devices and representational systems can differ and be combined in order to accomplish the task. Presence of multiple modalities involves fusion at every level. The case of emotion recognition is a special case of multimodal interaction. We consider passive recognition of emotions, which are highly synchronized responses to a stimulus. As such, multimodal fusion in our case doesn't involve syntactic or macro-temporal fusion as in [12]. We only consider micro-temporal fusion of data flows, thus reducing the problem of data fusion to a problem of synchronization.

In order to better integrate with works in multimodality for interactive applications [6], our implementation is inspired by the conceptual ICARE model. ICARE defines component types for devices and representational systems in the frame of interactive applications. The ICARE model is fully described in [15]. Our model is a specification of the ICARE model that defines a component type for devices and a component type for representational systems. Modalities combinations are handled by specific component types.

3.2. The Capture Unit component

3.2.1 Definition

A capture unit provides an interface with a capture device (e.g. Video camera, microphone, Electroencephalogram or Electromyogram sensors, motion tracking devices...). Its output is typically the measured signal but a capture unit can also involve some heavy processing, for example for extracting a human body's motion through video cameras. In this case, motion information will be the output of the capture unit. As an interface with a device, the capture unit component type is not specific to emotion recognition.

3.2.2 Consistency with multimodality

As such, the capture unit component type is fully identified as the device component type in the ICARE model.

3.3. The Feature Extractor component

3.3.1 Definition

The feature extractor's role is to analyze incoming data flows to extract one or several emotionally-relevant cues. A feature extractor is a step toward a higher level of abstraction. It can analyze captured flows or lower-level features flows. The outputted data types can differ greatly, from low-level cues (e.g. Value of energy of a human body at a time t) to high level features (e.g. computing the directness of a gesture after movement segmentation).

3.3.2 Consistency with multimodality

In the frame of multimodality, we identify the "feature extractor" component type to inherit from ICARE's "representational system" component type. Some properties are hence fixed. As a representational system, a feature is not arbitrarily chosen, as it is carefully identified as conveying emotional information. We chose to set the linguistic property of an interaction language to *false* as we are not aware of literature considering emotion expression as a structured language. Finally, we emphasize the importance of the temporal dimension of a feature. Static features can be computed at every frame and thus only need a fixed-size buffer to be computed: for example, distance between wrists can be computed at each frame from body coordinates. Accelerations of a torso movement can also be computed at each frame using a three-frame buffer. Dynamic features are computed over a varying period of time. For example, to compute the directness of a movement, one has to wait until the end of this movement.

3.4. The Interpreter

3.4.1 Definition

The interpreter's role is to analyze the value of a set of cues in order to infer an emotion. An interpreter can be represented as a function

$$f_{C\text{-->}E}(\{p\})$$

where C is the set of extracted features that will be analyzed for the interpretation; E is the set of studied emotion and their model (e.g. discrete set, continuous space, componential model); *f* is the interpretation function which, from the values of features in *C*, will deliver an emotion from *E*; *{p}* is the set of parameters for *f*.

An interpreter is an ad hoc component in the way that it is primarily shaped by the model and theory of emotions that serves as a basis for interpretation. This choice will condition the interpretation function *f*. In the typical case of a discrete model of emotions, *f* is usually a decision algorithm. The way that an interpreter is coded might then lead to increased modifiability in the input and output sets and in the parameters of function *f*.

3.4.2 Consistency with multimodality

We identify an interpreter as a representational system in the theory of multimodality. The properties of an interpreter are hence inherited from the "representational system" component type in ICARE. Due to the specificity of the emotion recognition domain however, we identified five properties specific to the "interpreter" component type.

Property 1. The chosen model of emotion. It conditions the available choices for interpretation function f. There are many theories and models of emotions but mainly three are present in the affective computing field: discrete models, continuous models, and componential models.

Property 2. Chosen set of delivered emotions and output format: Emotions are usually recognized among a predetermined set. The format in which they are delivered varies with the chosen emotion model, e.g. Words for discrete models, or coordinates for continuous spaces.

Property 3. Interpretation algorithm and its parameters: conditioned by the chosen emotion model, the algorithm can be a decision algorithm such as a neural network, rule-based system.

Property 4. Temporal dimension: As for feature extractors, interpreters can be static or dynamic. An interpreter relying on at least one dynamic feature is considered dynamic, as the dynamic feature may block the interpretation during its extraction.

Property 5. Considered cues: this property describes the features on which the interpretation is based.

3.5. Adapters and concentrators components

3.5.1 Definition

Adapters and concentrators are system-oriented component types. Adapters and concentrators are ad hoc components. Adapters function as an interface between two task-related components. Their role is to transform and adapt a data flow format. They can also be used to adapt the output data from a third-party application in order to plug it into an existing system based on the emotion branch model. Adapters hence allow better integration, reusability and modifiability with minimum tailoring. Concentrators' role is to merge data flows of the same type. For example, concentrators are used for merging data from two similar devices to increase robustness.

3.5.2 Consistency with multimodality

In terms of modality, adapters allow a transfer between two representational system. Contrary to the case of task-related component types however, this transfer does not trigger an increase in abstraction. Concentrators merge two data flows of a same type. This allows multiplying the sources for a signal, feature, or emotion flow, with the aim of increasing the robustness of the system. This corresponds to data fusion, in the sense of the signal processing domain. However, as the term "data fusion" may bear two meaning depending on the fields of signal processing and multimodal interaction. We hence chose the term concentrator to remove this ambiguity. Concentrators allow to handle equivalent and/or redundant representational systems. They should be developed to allow both using several flows to increase robustness and switching from a flow to another when needed.

4. Underlying mechanisms

The mechanisms described in this paragraph allow handling an assembly of components from the types described above. Those three mechanisms handle connections between the component, storage of the produce data, data synchronisation and memory management at running time.

4.1. The sequencer

In order to handle the components which types are described above, we developed a software engine "the sequencer" - which role is to centralize the acquired and produced data flows and to synchronize them. Within the system, data flows are composed of data blocks. Data blocks convey information at a time t, and have common properties. The sequencer is composed of tracks that are aligned over a timeline; each track corresponds to a data flow. A track then stores blocks from its corresponding flow along the timeline. When data is acquired from the world, a timestamp is applied to the corresponding block. This timestamp will be copied to every data block obtained via processing of this capture data block. This way, a computed emotion feature or emotion can be temporally aligned with the capture data it was extracted or interpreted from. Blocks are aligned in this manner along the timeline and one the various tracks. The sequencer handles data blocks and does not need to know about the conveyed information. This allows designing generic handling algorithms. The sequencer's data structure (list of tracks) can be hence stored in a XML file, thus completely decoupling structure data and handling algorithms. Apart from storing data blocks and aligning them along the timeline, the sequencer has two other tasks: synchronizing the data blocks before sending them to a component and managing the stored block to erase the useless ones. A block is considered useless when it has been consumed by every component that needed it as an input.

4.2. Synchronization pots

A synchronization pot is related to a component. It is a smaller version of the sequencer. It features a timeline and a track for each data flow the related component subscribed to. As such, if components are able to communicate the data flows they need as an input, synchronization pots can be created at execution time. A synchronization pot monitors the tracks that host the data flows needed by its related component. Each new block placed in a monitored track is copied in the synchronization pot, along the timeline. Once every track in the synchronization pot contains at least one data bloc, the whole block is sent to the related component. Synchronization pots allow getting rid of two issues in synchronization: it handles flows with different frequencies (length of a block) and phases (different offsets). Components, however, must be tailored to handle synchronization pots, as the number of block in each track may vary from frame to frame. Instantiating a synchronization pot for each component allows preventing blocking the whole system when a data flow fails to deliver information.

4.3. Garbage collector

The second algorithm featured in the sequencer is the garbage collector. As data blocks are stored in the various tracks of the sequencer, the memory cost of the system grows linearly. The garbage collector hence monitors the tracks in the sequencer and keeps track of each block's consumption. When a block has been consumed by every component that subscribed to it, it is erased.

5. Example: eMotion application

We illustrate our approach and the conceptual model described above by describing how we developed the application e-motion, based on the recognition of the emotion conveyed by a dancer.

Our computer-based gestural emotion recognition system relies on the component types described above. As we do not focus on identifying new expressive movement cues for emotion recognition, we drew characteristics from [7] in our emotion recognition system.

The system is composed of one capture unit: the Moven application, which provides an interface with the commercial motion capture suit Moven, from Xsens. From the flow of coordinates given by the Moven application, the eMotion software computes trunk and arm movement, vertical and sagittal directions, and velocity. Each feature is computed by a specific feature extractor component. The system then involves an interpreter component. The interpretation is then performed by choosing the maximum weighted sum of each cue over each of the six basic emotions. The eMotion software works at a frame level and delivers an emotion label at each frame. Emotion over a period of time is computed as the maximum in the ratios between the number of frames detected as a particular emotion and the total number of frames. The software was developed using TrollTech's Qt library, thus making the applications OS-independent.

With such a system, switching from the current motion capture suit to another only implies creating a new capture unit. Provided its output matches the Moven software output, nothing else has to be changed. Adding computer vision-based emotion analysis involves developing capture units for the cameras and specific feature extractors. Those new components can be plugged within the system and plugged directly to the existing interpretation component, provided the feature data flows match the inputs of the current interpreter.

6. Conclusion

In this paper we have presented a modifiable architecture model for emotion recognition software that integrates itself within the frame of multimodality in interactive applications. We presented the emotion branch and its five component types: the capture unit, the feature extractor, the interpreter, the adapter and the concentrator.

Each of the task related component-type can be instantiated as a simulated component. For example, a component can randomly deliver values for a data flow, or values according to some rules. A capture component will hence deliver a simulated signal, a feature extractor a simulated flow of characteristics, an interpreter a flow of emotions. A simulated component can also be driven by a human through the use of a graphical interface. This allows easy integration of software for a Wizard of Oz testing of a developed system. Of course, a human tester could better simulate feature extraction and interpretation than data capture. Finally, a component type instance can encapsulate a whole application for easier integration. For example, a feature extractor can encapsulate monolithic third-party software that extracts cues that are considered useful. An interpreter can encapsulate another emotion recognition system. This allows easy integration of a third-party software into the system, the only need being formatting the third-party software output interface to the corresponding component-type specifications.

Considered future works include the development of a software platform that would integrate existing toolkits for multimodal interactive applications and would offer graphical editors for assembling the components.

References

[1] A. Benoit, et al., Multimodal signal Processing and Interaction for a Driving Simulation: Component-Based Architecture. In *Journal on Multimodal User Interfaces*, 2007, Vol. 1, No. 1, Springer, pp. 49-58.

[2] R. Bolt, Put that there: voice and gesture at the graphics interface, Computer graphics, 262-270, 1980.

[3] A. Camurri, I. Lagerlof, and G. Volpe. Recognizing emotion from dance movement: comparison of spectator recognition and automated techniques. International Journal of Human-Computer Studies 59(1-2):213-225, 2003.

[4] A. Camurri, B. Mazzarino, G. Volpe Analysis of expressive gestures in human movement: the EyesWeb expressive gesture processing library, in Proc. XIV Colloquium on Musical Informatics, Firenze, Italy, May 2003.

[5] G. Castellano, S.D. Villalba, and A. Camurri: Recognizing human emotions from body movements and gesture dynamics. Affective computing and intelligent interaction, 71-82, Springer Berlin-Heidelberg, 2007.

[6] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R. Young. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties In Proceedings of the INTERACT'95 conference, S. A. Arnesen & D. Gilmore Eds., Chapman&Hall Publ., Lillehammer, Norway. pages 115-120. 1995.

[7] M. De Meijer. The contribution of general features of body movement to the attribution of emotions. Journal of Nonverbal Behavior, 13(4):247-268, 1989.

[8] S. D'Mello, R. W. Picard, and A. Graesser. Toward an Affect-Sensitive AutoTutor. *IEEE Intelligent Systems* 22, 4 (Jul. 2007), 53-61.

[9] G. Krasner, S. Pope. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. In *Journal of Object Oriented Programming*, 1988, Vol. 1, No. 3, pp. 26-49.

[10] F. Nasoz, O. Ozyer, C. Lisetti, N. Finkelstein. Multimodal affective driver interfaces for future cars. In *Proc. Of Multimedia'02*, December, 1-6, 2002, France, ACM, pp. 319-322.

[11] L. Nigay, Modalité d'interaction et multimodalité, Université Joseph Fourier, 2001.

[12] L. Nigay and J. Coutaz. *A design space for multimodal systems : concurent processing and data fusion*. Proc. of INTERCHI'93. Amsterdam, april 24-29, 1993, ACM Press. pp 172-178.

[13] K. R. Scherer, Emotions as episodes of subsystem synchronization driven by nonlinear appraisal processes, Emotion, Development, and Self-Organization, Cambridge University Press, New York/Cambridge, p.70–99 (2000)

[14] The UIMS Tool Developers Workshop, A Metamodel for the Runtime Architecture of an Interactive System. In SIGCHI Bulletin, 1992, pp. 32-37.

[15] J. Bouchet and L. Nigay, ICARE: *A Component-Based Approach for the Design and Development of Multimodal Interfaces*. Proc. of ACM-CHI'04. Austria, april, 2004, ACM Press, pp. 1325-1328.

[16] A. Clay, N. Couture, L. Nigay *Towards an architecture model for emotion recognition in interactive systems: application to a ballet dance show. i*n Proc. WINVR09, Chalon-sur-Saône, France, February 25-26, 2009.