rapidmix

# D4.1 Report on Candidate Technologies and APIs to be developed

| | |
|---|---|
| Grant Agreement nr | 644862 |
| Project title | Realtime Adaptive Prototyping for Industrial Design of Multimodal Interactive eXpressive technology |
| Project acronym | RAPID-MIX |
| Start date of project (dur.) | Feb 1st, 2015 (3 years) |
| Document reference | RAPIDMIX-D-WP4-RS-150930-ReportOnCandidateTechnologies-1.7 |
| Report availability | PU - Public |
| Document due Date | September 30th, 2015 |
| Actual date of delivery | September 30h, 2015 |
| Leader | RS |
| Reply to | Andrés Bucci (andres@reactable.com) |
| Additional main contributors (author's name / partner acr.) | |
| Document status | Final document |

Programme co-funded by the European Commission

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 644862

## *EXECUTIVE SUMMARY*

The main goal of the RAPID API is to provide powerful but easy to use toolkit for integrating novel sensor technologies into products and prototypes within the creative community. For this to be completed successfully we want to test and compare the different technologies within the consortium to be able to integrate them into a coherent tool with the widest possible application spectrum.

This deliverable contains a report that helps to identify key elements and characteristics for each of the technologies that will be integrated to the API. It feeds from each partner's input and its contents and approach were also discussed in the working meetings (both in Paris June 20-22 and London September 8-9). Its contents should be one of the relevant tools to create the first version of the fully functional API, as well as providing insight on possible improvements from each of the technology providers inside the consortium.

## BACKGROUND

This document is the first line of work in the process of developing the RAPID-API. It should inform the list of early prototypes to build in T3.1 and the first UCD Workshops on T2.2. It aims at categorizing and comparing different candidate technologies and proofs of concept in order to evaluate the relevant functionality that can be offered to future users of the API.

*Table of Contents*

1 INTRODUCTION ..... 5

1.1 Main objectives and goals ..... 5

1.2 Structure of the document ..... 5

2 METHODOLOGY ..... 6

2.1 Categorization and Comparison ..... 6

2.1.1 Interactive Machine Learning and Gesture Interpretation ..... 6

2.1.2 Online repositories and collaborative communities ..... 9

2.1.3 Real-time sound synthesis ..... 9

2.1.4 Software development platforms ..... 12

2.1.5 Sensors and hardware platforms ..... 12

2.1.6 Music Information Retrieval (MIR) and feature extraction ..... 14

2.2 Testing ..... 16

3 CONCLUSION ..... 18

4 REFERENCES ..... 19

4.1 Scholar references ..... 19

4.2 Web references ..... 19

4.3 Acronyms and abbreviations ..... 19

# *1 INTRODUCTION*

## 1.1 Main objectives and goals

The main goal of this document is to provide a description and analysis of the partners' PoCs and candidate technologies, which will have been tested, compared, and classified according to their application domains and their Technology Readiness Levels.

## 1.2 Structure of the document

This deliverable is structured as follows:

- We define a methodology for categorizing the different candidate technologies and some comparison criteria specifically tailored for each of the categories.
- Based on that methodology we report key aspects of each technology present from within the consortium.
- We discuss the methodology to test the candidate technologies in their current state and provide information, when available, of their trajectory within the project.
- Finally, we draw conclusions that will help inform the requirements for the first version of the API, along with the work carried out and reported in the first Design Specifications document (D2.4) and the early prototypes (D3.1).

# 2 METHODOLOGY

According to the DoW there already is a wide variety of candidate technologies within the consortium that are useful for many applications within the creative industries. For an API to be developed and used successfully, we must first identify the areas in which our existing tools will make the development of new products easier and provide a way to connect different inputs and processes in a coherent and useful manner; we must also identify key factors that make the candidate technologies best suited for a specific task.

The approach followed in this report will be then to first categorize and compare the different technologies according to their application domain and defining relevant criteria to compare them. Then we will discuss the criteria for testing the initial versions of those technologies and extract some information that could be of use in the design of a first version of the RAPID-API.

## 2.1 Categorization and Comparison

Candidate technologies will be grouped according to functionality and application domain in order to establish possible overlaps and unique features of each member in the consortium. The comparison criteria will be specific for each of the categories, and a brief summary of the findings in each category will be given, when not entirely self explanatory.

### 2.1.1 Interactive Machine Learning and Gesture Interpretation

Within this category we find machine learning technologies that allow the user to interactively demonstrate examples of the desired output on a training phase and then interpret a new unknown example by classifying it, or providing an output that is proportional to it. They allow for fast training and continuous, real-time inference. Inside the consortium the technologies that belong to this category are:

**XMM**
XMM is a cross-platform C++ library that implements Gaussian Mixture Models and Hidden Markov Models for recognition and regression. The XMM library was developed for movement interaction in creative applications.

**Gesture Follower (GF)**
Gesture Follower provides probabilistic estimations of the similarity of the performed gesture to pre recorded gestures (likelihood) and the time progression of the performed gesture. It then allows for the estimation of the current temporal index inside the gesture, referred here as “gesture following”, aside from recognizing it from the provided examples. It is specially targeted to synchronizing continuous visual or sound processes to gestures.

**Gesture Variation Follower (GVF)**
Gesture Variation Follower is a C++ library that provides methods for estimating variations (e.g. in scale or orientation) of detected gestures from a set of examples. It is also targeted for creating creative applications such as controlling sounds and visuals.

**GestureAgents**
GestureAgents is a framework for building collaborative multi-user systems with support for concurrent multi-tasking and shareable interfaces running simultaneously in different applications. It allows several applications to share a single input source for gestures, preventing double-consumption of input events, while allowing free independent implementations of gesture recognizers, meaning they don’t have to share a single gesture recognition mechanism.

**Wekinator**
Wekinator is a software package for designing interactive systems aimed at live music performance and other real-time domains using a collection of machine learning algorithms. It allows users to design interactions by demonstrating human actions and computer responses, instead of writing programming code.

#### *2.1.1.1 Comparison*
The comparison criteria chosen for the technologies inside this category is the available machine learning algorithms, the type of software product, the implementation language and the licensing information available. These results can be observed in the table below.

We can observe that while the application domain remains similar for all the identified technologies, since they were conceived to work in the interactive application and musical domains, we have a clear separation between software components or libraries that implement standard machine learning algorithms, such as XMM and Wekinator and others that provide their own algorithms for classification, such as GestureAgents and both GF and GVF. There seems to be some areas of overlap regarding the types of classifiers implemented within the standard machine learning algorithms, and probably the implementation language unification and the removal of license restrictions could be key factors in a future version of the API.

| | **XMM** | **GF** | **GVF** | **GestureAgents** | **Wekinator** |
|---|---|---|---|---|---|
| **Available algorithms** | Gaussian Mixture Models (GMM)<br>Gaussian Mixture Regression (GMR)<br>Hierarchical Hidden Markov Model (HHMM)<br>Hierarchical Hidden Markov Model Regression (HHMR)<br>Simple template or multiple example | Specific HMM implementation using single templates for time alignment and recognition | Dynamic System with Particle Filtering using templates for continuous parameter evaluation | GestureAgents algorithm, based on the Agent Exclusivity mechanism. | Supervised classifiers (AdaBoost, SVM, kNN, j48, Naive Bayes)<br>Regression (multilayer perceptron, linear & polynomial regression)<br>Dynamic Time Warping |
| **Type of software** | Library | Library | Library | Library | Standalone desktop application with a Graphical User Interface (GUI). It currently also has an Open Sound Control API that allows it to be controlled from other applications |
| **Implementation language** | C++ | C++ | C++ | Python | Java |
| **Licensing information** | GPLv3 | Proprietary | LGPLv3 | MIT | GPLv2 (because of Weka library dependency) |

### 2.1.2 Online repositories and collaborative communities

Within this category we include all the different tools that allow online storage and visualization of large collections of data.

**Freesound**
Freesound is a collaborative online sound database distributed under the creative commons licenses. It provides several ways of accessing its sound files, allowing users to browse the sounds using keywords, a "sounds-like" type of browsing; uploading and downloading sounds to and from the database. It also provides ways to interact with other members of the community. It aims to create an open database of sounds that can also be used for scientific research.

**RepoVizz**
RepoVizz is an integrated online system capable of structural formatting, remote storage, browsing, exchange, annotation, and visualisation of synchronous multimodal data. It has a powerful HTML5 visual interface which is accessible from any standard web browser, making it platform-independent.

#### *2.1.2.1 Comparison*

To compare these two technologies, we focus on the Type of API they provide third-party users to interact with the hosted data as well as in the type of data they store.
We can observe that Repovizz and Freesound are repositories for different types of data, and they do not substantially overlap. They both have RESTful APIs that could be easily accessed from the RAPID-API.

| | **Freesound** | **RepoVizz** |
|---|---|---|
| **API Type** | REST | REST |
| **Stored data types** | Audio, extracted audio features | Time-aligned streams of heterogeneous data: audio, video, motion capture, physiological signals, extracted features, annotations |

### 2.1.3 Real-time sound synthesis

In this category we have the technologies that have capabilities of generating a continuous stream of sound output, regardless of the methods used to produce them.

**Interactive Audio Engine (IAE)**
IAE is a collection of software libraries designed for content based audio processing. The engine can extract audio descriptors from recorded audio materials and provide asynchronous/synchronous granular synthesis and additive synthesis.

**Maximilian**
Maximilian is a cross-platform sound synthesis library designed for simplicity of use. It provides additional sound processing functionality for common effects in music such as delay, reverb, flanger, chorus and distortion, as well as multi-channel mixing, audio sample recording, and playback.

**Collaborative Situated Media (COSIMA)**
COSIMA is a set of components for audio processing, motion analysis and collective interaction based on web/mobile technologies (HTML5, Javascript, and Web Audio API).

#### *2.1.3.1 Comparison*

The most relevant comparison criteria for these tools would be the types of synthesis algorithms that they offer to the users; however other relevant aspects are if they can also provide sound analysis capabilities, their implementation language, the licensing information, the software dependencies and platform support. These results can be observed in the table below.

The first observable quality within this category is the fact that most of the standard sound synthesis techniques are covered with the existing technologies inside the consortium, there seems to be no evident gap or missing feature, and there is a strong focus on concatenative and granular synthesis; however, there seems to be some overlap between the techniques in IAE and Maximilian on the most basic types of synthesis, as would be expected. There is a clear distinction on the purpose of COSIMA with respect to the other members of this category as it is intended to run within the browser and not to be run as a library. A key aspect to note for future versions of the API would be to take into account the restrictions imposed by the closed source members.

| | **Interactive Audio Engine (IAE)** | **Maximilian** | **COSIMA** |
|---|---|---|---|
| **Available synthesis algorithms** | Additive synthesis<br>Granular synthesis<br>Concatenative synthesis<br>Overlap-add synthesis | Additive synthesis<br>Granular synthesis<br>Subtractive synthesis<br>Wavetable synthesis<br>Spectral synthesis<br>Sample playback<br>Atom synthesis | Granular synthesis<br>Concatenative synthesis |
| **Sound analysis capabilities** | Audio and voice analysis modules<br>Spectra and statistical measures on spectra<br>Mel-frequency cepstral coefficients (MFCC)<br>MEL and Bark coefficients<br>Cepstrum<br>Pitch, Periodicity (Noisiness), Loudness<br>LPC | Spectral analysis (FFT)<br>Log and linear magnitude power spectrum<br>Constant Q transform (configurable)<br>Peak Frequency (pitch)<br>Spectral centroid<br>Spectral flatness<br>MFCC<br>Bark scale analysis | |
| **Implementation language** | C++ | C++ | HTML5, Javascript, Web Audio API |
| **Licensing information** | Closed source | MIT | Closed Source |
| **Platform support** | Mac<br>Windows<br>iOS | Mac<br>Windows<br>Mac<br>Linux<br>Android<br>iOS or ARM embedded | Multiplatform (web browser with HTML5) |

### 2.1.4 Software development platforms

**JUCE**

This category has only one candidate technology, JUCE, which does not have a specific use but rather is intended to provide a wide-range of solutions for common implementation tasks when developing applications. In this case instead of comparison criteria we will provide key elements in which this technology has a relevant difference with respect to its competitors, where applicable. The most popular cross platform frameworks that can also be compared with JUCE are Qt and Unity in the finance, automotive and game industries; Cinder in the interactive graphics domain; OpenFrameworks and Processing in the multimedia platform and creative coding industries.

#### *2.1.4.1 Comparison*

With respects to the above mentioned frameworks we can establish that JUCE has certain comparable aspects, such as its extensibility and modular architecture, a wide range of possible application, cross-platform support for the major operating systems. The key feature in which we found it stands out from its competitors is that it is already widely used within the audio industry, especially in the creation of music plugins in different formats.

### 2.1.5 Sensors and hardware platforms

In this category we will include all the hardware platforms which contain sensors that will allow analog or digital signals coming from different physical properties to be processed or communicate them to other software components.

**BITalino**

BITalino is a low-cost toolkit to learn and prototype applications using body signals. Targeted for a wide range of users (students, teachers, makers, artists, researchers, corporate R&D) that have little or no electrical skill.

**Gestureplux**

Gestureplux is a wearable device that allows hands free control of mobile or PC devices by interpreting hand gestures coming from EMG and accelerometer data.

**HapticWave**

Hapticwave is a prototype to enable audio engineers with visual impairments to have quicker and more efficient interactions with computer-based recording and editing of audio, using haptic feedback.

**EAVI IoT PCB**

The IoT PCB is a multi-parametric wireless, palm mounted, low-profile wearable interface for the control of a computer system or mobile device through a group of sensors. It has two main components that can be used individually or together.

#### *2.1.5.1 Comparison*

The most relevant comparison criteria for this category would be the information regarding specific hardware concerns such as input/output capabilities, sampling rate, connection type, and chip architecture. The results can be seen in the table below.

The first observable quality from these technologies is that some of them seem to be targeted for a very specific use or type of signal (HapticWave and GesturePlux), while some try to be more generic in their purpose. We can also identify products that are closer to prototypes, and mature products that are already being commercialized. There seems no be no overlap and a clear distinction of the functionalities, as well as a strong focus on movement and biosignals instead of a more generic prototyping hardware approach. Aside from HapticWave, their small size makes them ideal candidates for products requiring wearable technology applications.

| | **BITalino** | **Gestureplux** | **HapticWave** | **EAVI IoT PCB** |
|---|---|---|---|---|
| **Input** | 4 (10 bit) analog inputs<br>2 (6 bit) analog inputs<br>4 (1 bit) digital inputs | EMG (2 channels)<br>Accelerometer | 1 Analog input | Accelerometer<br>Gyroscope<br>Digital inputs |
| **Output** | 4 (1 bit) digital outputs | Software data streams<br>Software event detection | Haptic force | Software data streams |
| **Sample rate** | Configurable to 1, 10, 100, or 1000Hz | 800Hz | N / A | N / A |
| **Connection type** | Class II Bluetooth v2.0 | Bluetooth low energy | USB | Bluetooth low energy |
| **Architecture** | Atmel AVR ATMega Chipset | Atmel AVR ATXMegachipset | AVR Microcontroller | 8051 Microcontroller |

### 2.1.6 Music Information Retrieval (MIR) and feature extraction

In this category we will list the different technologies capable of deriving informative, non-redundant information from a stream of data, in the audio and biosignal domains. The IAE and Maximilian include feature extraction as part of their other main features so they will not be discussed in this section.

**Teclepathy**
Teclepathy is a framework that allows the acquisition of affective, attentive, and cognitive states from the human body using wearable, non-invasive hardware. Targeted for defining control mappings for sonic interaction design.

**Maven**
Maven is a statistical modelling and machine-learning based framework for the creation of context-aware features for information visualisation, and audiovisual scene representation. It allows for feature models to be constructed that better represent audio and visual information based on how the information is perceived. It is targeted at visualization of complex auditory scenes and large audio datasets.

**Essentia**
Essentia is a cross-platform C++ library for audio analysis and audio-based music information retrieval. It contains an extensive collection of reusable algorithms which implement audio input/output functionality, standard digital signal processing blocks, statistical characterisation of data, and a large set of spectral, temporal, tonal and high-level music descriptors.

#### *2.1.6.1 Comparison*

The technologies in this category will be compared by the types of feature extractors they offer and on the types of signals these feature extractors were designed to work, as well as their license information and software dependencies, their implementation language, and their platform support.

Within this category there also seems to be a great number of both high and low level descriptors that could be available to users of a future version of the API, and we can observe that there is a strong focus on the audio descriptors, with just Teclepathy being specifically designed to provide EEG features. There seems to be some overlap between Maven and Essentia in the onset and beat detection descriptors, although Maven is not restricted to audio signals. All of the technologies listed above are cross platform and mostly implemented in C++, this should prove an advantage in the future for product integration, but maybe some considerations should be made to make versions that make it easier to quickly prototype applications in a less strict language.

The only limitation we can observe within this group is the dependency on a third party software such as Matlab and pure data in Teclepathy that could prove to be a deterrent when trying to develop future applicaitons; licensing considerations should also be taken into account to provide the maximum usability for future adopters of the API, specially when providing access to the descriptors available only in Maven.

| | **Teclepathy** | **Maven** | **Essentia** |
|---|---|---|---|
| **Feature extractors** | EEG frequency band analysis for extracting distinguishable band features. | Onset detection<br>Beat detection<br>Song segmentation<br>Novelty detection<br>Interest measures<br>Attention estimation<br>Visualisation filtering | Statistical descriptors<br>Time-domain descriptors<br>Spectral descriptors<br>Tonal descriptors<br>Onset detection<br>Beat detection<br>High-level descriptors (danceability, song segmentation, dynamic complexity)<br>Different types of filters and other standard types of signal processing blocks |
| **Type of signal for feature extraction** | EEG oscillatory activity | No specific type | Audio signals |
| **Licensing information** | Closed source | Closed source | Affero GPLv3 license |
| **Implementation language** | C++<br>Matlab<br>Pure data | C++ | C++ |
| **Platform support** | Cross-platform | Cross-plaform | Cross-platform |

## 2.2 Testing

The methodology for testing the different candidate technology was jointly decided by the consortium during the May 2015 working meeting that took place in IRCAM, Paris, where key attributes that could be monitored for each of the candidate technologies in their current state would be reported, and their possible states within one year and at the end of the project. These key features were identified as:

**Programmability**
The ability of the user to extend the provided software product. This was regarded as a desirable characteristic on the existing software components and to observe whether or not it can be added as a feature in the duration of the project.

**Possible extension points**
The current and foreseen extensions that could be developed during the course of the project. This characteristic should inform of possible features to be implemented or natural common goals that could be developed for several technologies.

**Input and output data**
The types of data that each candidate technology is able to receive, or provide to another application as a result of its internal processes. This helps identify possible issues with data formats or characteristics of certain technologies, whether they accept streams of data or process the data offline.

**Processing capabilities**
This category was selected to observe the individual tasks performed by each of the technologies and to monitor their development within the duration of the project.

**System requirements**
Any hardware components or software resources that need to be present in the system that runs the candidate technologies. This was reported to observe possible blocks or restrictions that would make a future API useful for the largest possible audience.

**Platform**
Operating system on which the developed software components are designed to run. This could inform of possible limitations for certain technologies and signal possible development paths to extend their functionality.

**Architecture**
Regarding the hardware platforms, this category was selected to monitor difference between different instruction sets present in the various microcontroller families.

**TRL**
An initial assessment of the technology readiness level was reported by each partner in order to monitor their development and ensure the expected level at the end of the project.

The detailed results of this report can be seen in Appendix 1. In addition, a brief comparison with existing technologies outside the consortium was also identified as an important value to observe in order to quickly identify possible areas of strength or improvement.

Licenses and software dependencies relating to all RAPID-MIX technologies are being examined by an independent third party lawyer as part of the IPR report which is part of work package 1 (D1.3, due on

January 2016), as a result, these details will be reported on as part of that specific deliverable and will not be included in this report.

From the compiled data we can already make several observations, for example that only a few of the candidate technologies have a clear and planned development roadmap. This can be seen as an opportunity to integrate in a coherent manner with other partner's technologies and quickly integrate them into a future version of the API; we can also observe that there seems to be already a wide coverage for the major operating systems and platforms, and that the TRL levels reported for each technology are on average beyond the point of demonstration in a relevant environment, with some of them already featuring in commercial products, and that there is a clear intention of raising this level within the whole consortium.

The comparison with existing technologies is also an indicator of specific areas of improvement or differentiation, and we can observe that there is no evident factor in which there needs to be a significant amount of work to match or exceed their performance.

# 3 CONCLUSION

We have provided relevant information from every partner's technology to categorize them according to relevant criteria that allowed key features to be compared. A brief summary of the results of these comparisons was presented in order to help identifying the next steps to be taken when designing the future versions of the RAPID API.

Gathering this information also revealed other practical aspects that need to be addressed within the duration of the project, such as licensing restrictions and implementation language.

# 4 REFERENCES

## 4.1 Scholar references

Bogdanov, D., Wack N., Gómez E., Gulati S., Herrera P., Mayor O., et al. (2013). *ESSENTIA: an Audio Analysis Library for Music Information Retrieval*. International Society for Music Information Retrieval Conference (ISMIR'13). 493-498.

Schnell, N., Röbel, A., Schwarz, D. Peeters, G., & Borghesi, R. (2009). *MuBu & Friends - Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP*. International Computer Music Conference (ICMC). Montreal.

Grierson, M., Kiefer, C. (2013) *NoiseBear: A Malleable Wireless Controller Designed In Participation with Disabled Children*. In Proceedings of the International Conference on New Interfaces for Musical Expression, Graduate School of Culture Technology, KAIST.

## 4.2 Web references

Music Technology Group, Universitat Pompeu Fabra (2013). Repovizz [website]. Retrieved from https://youtu.be/c7pmDvkKY7A.
ROLI Ltd (2015). JUCE [website]. Retrieved from http://www.juce.com/features

## 4.3 Acronyms and abbreviations

PoCs – Proofs of Concepts
IoT – Internet of things
EAVI – Embodied Audiovisual Interaction Group
DoW – Description of Work
MIR – Music Information Retrieval
FFT – fast Fourier transform
LPC – Linear predictive coding
EMG – Electromyography
EEG – Electroencephalography
PCB – Printed circuit board

## *APPENDIX 1: Table of Technologies*

The following table was used in the testing of each candidate technology and was provided by each partner.

**BITALINO**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Yes (but at low level in C) | Yes (but at low level in C) | Yes (Arduino-style C) | + More optimized for performance<br>- Less user-friendly reprograming |
| ***Possible extension points*** | Supports third-party sensors with analog output | Supports third-party sensors with analog output | Supports third-party sensors with analog output; SPI interface | + Analog interfaces are easier to use (plug & play)<br>- Nowadays many third-party accessories only have digital interfaces |
| ***Input data*** | Up to 4 (10-bit) and 2 (6-bit) analog inputs + 4 (1-bit) digital inputs; Sensors bundled by default are ECG + EMG + EDA + Accelerometer + Light (expansible with third-party sensors) | Up to 4 (10-bit) and 2 (6-bit) analog inputs + 2 (1-bit) digital inputs; Sensors bundled by default are EEG + ECG + EMG + EDA + Accelerometer + Light + Pushbutton (expansible with third-party sensors) | Up to 4 (10-bit) and 2 (6-bit) analog inputs + 2 (1-bit) digital inputs; Sensors bundled by default are EEG + ECG + EMG + EDA + Accelerometer + Light + Pushbutton (expansible with third-party sensors) | + Provides good compromise between functionality and flexibility<br>- Has less analog inputs than comparable professional systems |
| ***Processing capabilities*** | No | No | Yes (optional) | + Enables better energy and operational efficiency<br>- Less flexible than competing technologies |
| ***Output data*** | 4 (1-bit) digital outputs; Actuator bundled by default is an LED | 2 (1-bit) digital outputs + 1 (10-bit) digital-to-analog converter; Actuators bundled by default are an LED and a Buzzer | 2 (1-bit) digital outputs + 1 (10-bit) digital-to-analog converter; Actuators bundled by default are an LED and a Buzzer | + Offers enough functionality for the average user<br>- Limiting for power users |

| | | | | |
|---|---|---|---|---|
| ***System Requirements*** | Bluetooth 2.0 | Bluetooth 2.0 | Bluetooth 2.0, BLE, or USB | + More practical due to default wireless connectivity<br>- Not as interoperable as USB for standard computer use |
| ***Platform*** | Win; Linux; Mac OS; Android | Win; Linux; Mac OS; Android | Win; Linux; Mac OS; Android; iOS | ± On par with competitors |
| ***Architecture*** | Atmel AVR ATMega chipset | Atmel AVR ATMega chipset | Atmel AVR ATMega chipset (eventually upgrade to ARM Cortex M0 or analogous) | + Based on a trialled, tested, and easy-to-use "work horse" chipset<br>- Has limited processing power to enable on-board processing |
| ***Technology Readiness Level*** | 9 | 9 | 9 | ± On par with competitors |

**Wekinator**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Java | | | + easy to maintain cross-platform code<br>+ compatible with robust machine learning library (Weka) for fast development & experimentation<br>+ Weka code can easily be taken out and replaced for less restrictive licensing<br>- won't run on iOS |
| ***Possible extension points*** | Takes input from anywhere, sends output to anywhere via OSC (can be rerouted to MIDI) | | | + extremely flexible: use as middleware in prototyping arbitrary applications with arbitrary hardware/software |

| | | | | |
|---|---|---|---|---|
| ***Input data*** | Anything that can be represented as real-time constant-length vector of floats (has been used with computer vision, game controllers, audio input, on-body sensors, Bitalino, Arduino, Kinect, etc.) A "WekiHelper" standalone app optionally adds signal conditioning, rate throttling, up/downsampling, etc. | | | + flexible and fast<br>- not everyone's code is already OSC enabled, or should be |
| ***Processing capabilities*** | 1) Supervised learning classification (AdaBoost, kNN, J48 decision trees, SVMs) 2) Supervised learning regression (neural nets, linear & polynomial regression), 3) Multidimensional temporal path recognition using fast dynamic time warping | | | + standard algorithms are useful for diverse applications<br>+ Weka (GPL) implementations can be swapped for equivalent functionality to gain fewer licensing restrictions |

| | | | | |
|---|---|---|---|---|
| ***Output data*** | For 1) and 2), outputs constant-length vector of floats via OSC. | | | + fleixble and fast<br>+ integrates with tools musicians are already using<br>- not everyone's code is already OSC enabled, or should be |
| ***System Requirements*** | Java 7 or higher | | | + runs on nearly every modern computer without special attention<br>- no iOS |
| ***Platform*** | Desktop | | | + good for prototyping (plenty of flexibility in making GUIs for experimentaiton, visualisation; machine learning is very fast)<br>- embedding into mobile apps can't currently be done out of the box (requires cannabalizing code) |
| ***Architecture*** | | | | see above |
| ***Technology Readiness Level*** | | | | + older version of Wekinator has been used in dozens of performances of new musical interfaces, has shown to be extremely useful as a prototyping tool for composers |

**Repovizz**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Java and JS/HTML/CSS | Javascript (for custom visualization modules) | Python (for processing/feature extraction modules) | + API exposed directly from the backend<br><br>- Front-end client architecture not yet standardized across different platforms |
| ***Possible extension points*** | Refactoring backend, to convert it into a framework/service | | | |
| ***Input data*** | Multimodal data + XML structure | Multimodal data + JSON structure | Multimodal (both real-time and stored) data + JSON structure | + Theoretically any data type can be stored and interpreted by the back-end<br><br>- Unconventional/less known data types require the explicit specification of a custom data type |
| ***Processing capabilities*** | Essentia | | | |
| ***Output data*** | Visualization, JSON summaries of datapacks and nodes, OSC streaming (prototype) | Visualizations, JSON summaries of datapacks and nodes, Real-time streaming of datapack contents | Visualizations, JSON summaries of datapacks and nodes, Real-time streaming of datapack contents, Computed descriptors/features | + Data can be served over HTTP and websockets |
| ***System Requirements*** | Google Chrome (for client), Java (for | Client-dependent | Client-dependent | |

| | | | | |
|---|---|---|---|---|
| | datapack designer) | | | |
| ***Platform*** | Web | Client-dependent | Client-dependent | |
| ***Architecture*** | Java (backend), JS/HTML/CSS (client) | Python (backend), JS/HTML/CSS (client) | Python (backend), JS/HTML/CSS (client) | + Data accessible from anywhere<br>- No support for local deployment yet |
| ***Technology Readiness Level*** | 4 | 6 | 9 | |

**Interactive Audio Engine (IAE)**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | C++ | | | no dependencies, include IRCAM libraires: RTAlib (sound analysis), MuBu (data structure), PiPo (plugin for stream processing), ZsaZsa (Granular synthesis, Concatenative) |
| ***Possible extension points*** | compatible with SndFile | integratiuon wih IMTReditor : data visualization, addlib: additive synthesis, specsyn: spectral synthesis | | |
| ***Input data*** | Audio + Multimodal Data (both real-time and recorded) | | | |
| ***Processing capabilities*** | Sound and Multimodal Analysis (Descriptors, Spectrum, Filtering), Sound Synthesis, Indexing (Kd Tree) | | | |

| | | | | |
|---|---|---|---|---|
| ***Output data*** | Audio + Multimodal Data (both real-time and recorded) | | | |
| ***System Requirements*** | Multiplatform: tested with OSX, Windows, iOS | | | |
| ***Platform*** | Multiplatform: tested with OSX, Windows, iOS | | | |
| ***Architecture*** | classes hiearachy, also with an C API (e.g. to be integrated in Unity 3D) | | | |
| ***Technology Readiness Level*** | 9 (already in Industrial License and commercial audio products | | | |

**Gesture Follower**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | C++ library | Improvement planned: improved the estimation of relative velocity, add contraints betweeen position and velocity | | Has no dependencies |
| ***Possible extension points*** | | | | |
| ***Input data*** | multidimentional data | | | can handle times series in real-time, |
| ***Processing capabilities*** | Time alignement and recognition based on Hidden Markov Models | | | simple learning procedure using a single example, few parameters to adjust |
| ***Output data*** | Time progression and likelihoods | | | continuous likelihoods update and time progression |
| ***System Requirements*** | Destop or Mobile | | | |
| ***Platform*** | Crossplatform, implemented in OSX, iOS, Windows, implemented in Max7 as an external (MuBu library) | | | the implementation in Max allows for using visualization tools with the MuBu library |
| ***Architecture*** | C++ library | | | |
| ***Technology Readiness Level*** | 8 | 9 | 9 | |

**Gesture Variation Follower**

| | Current State | 1 Year | 3 Years | Comparison with existing technologies |
|---|---|---|---|---|
| **Programmability** | C++ library | | | |
| ***Possible extension points*** | | | | |
| ***Input data*** | multidimensional data | | | can handle times series in real-time, |
| ***Processing capabilities*** | Recognition and Adaptation/Estimation of various parameters: time, scale, relative speed, orientation etc. | | | simple learning procedure using a single example, can adapt to gesture variations |
| ***Output data*** | Time progression, likelihoods, scales, relative speed, orientation etc | | | estimate in real-time the parameters such as time, scale, relative speed, continuous likelihoods update and time progression |
| ***System Requirements*** | Desktop or Mobile | | | |
| ***Platform*** | Crossplatform, currently implemented in Max7, PD and openFrameworks | | | |
| ***Architecture*** | C++ library | | | |
| ***Technology Readiness Level*** | 7 | | | |

**XMM**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | C++ | improvement of the API planned, extensive evaluation planned | | Has no dependencies except libjson that is optional to save models in json formt |
| ***Possible extension points*** | | | | |
| ***Input data*** | multidimentional data, e.g. movement and/or sound parameters | | | Can handle times serie in real-time, |
| ***Processing capabilities*** | Recogntion and Regression, Gaussian Mixture Models or Multimodal Hierarchiacal Hidden Markov Models | | | use of HMM for regression, hierachical structures, training from single template or several examples |
| ***Output data*** | Time progression and likelihoods, sound control prameters in case of regression | | | can generate directly sound parameters, continuous likelihoods update and time progression |
| ***System Requirements*** | Destop or Mobile | | | |
| ***Platform*** | Crossplatform, implemented in OSX, Windows, implemented as externals in Max7 (MuBu library) | | | the implementation in Max allows for using visualization tools with the MuBu library |
| ***Architecture*** | C++ library | | | |
| ***Technology Readiness Level*** | 7 | 8 | 9 | |

**COSIMA**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | yes (Javascript) | | | |
| ***Possible extension points*** | | | | |
| ***Input data*** | audio and control (movement) | | | |
| ***Processing capabilities*** | movement and audio processing, sound synthesis (granular and concatenative, oscillator, filtering) | important improvements and extensions will be added | expected to support a large number of reserach and industrial projects | |
| ***Output data*** | audio, visualization, processed data | | | |
| ***System Requirements*** | HTML5 with WebAudio API | | | |
| ***Platform*** | Mobile (HTML5 with WebAudio API) | | | |
| ***Architecture*** | | | | |
| ***Technology Readiness Level*** | 7 | 8 | 9 | |

**JUCE**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | C++ | | | |
| ***Possible extension points*** | Adding JUCE modules | | | |
| ***Input data*** | Audio, Midi, OSC, XML, JSON... | | | |
| ***Processing capabilities*** | Utility processing routines for Audio (FFTs, IIR), Midi and data | | | |
| ***Output data*** | Audio, Midi, OSC, XML, JSON... | | | |
| ***System Requirements*** | | | | |
| ***Platform*** | Windows, OS X, Linux, iOS, Android, Embedded | | | |
| ***Architecture*** | VST, VST 3, AudioUnit, AAX... | | | |
| ***Technology Readiness Level*** | 9 | | | |

**Gestureplux**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Yes (but at low level in C) | Yes (but at low level in C) | No | ± On par with competitors |
| ***Possible extension points*** | No | No | Memory card style swappable sensor dock | ± On par with competitors |
| ***Input data*** | EMG (2 channels); Accelerometer | EMG (2 channels); Accelerometer | EMG (2 channels); Accelerometer; ECG & Temperature (as optional sensor add-on) | + Provides good compromise between formfactor and functionality<br>- Has less inputs than comparable consumer products (e.g. Myo armband) |
| ***Processing capabilities*** | EMG onset detection; EMG activation duration; EMG activation frequency; 3-axial tilt | EMG onset detection; EMG activation duration; EMG activation frequency; 3-axial tilt | EMG onset detection; EMG activation duration; EMG activation frequency; 3-axial tilt; Heart rate; Peripheral temperature | ± On par with competitors |
| ***Output data*** | Events resulting from the processing; Raw data stream | Events resulting from the processing; Raw data stream | Events resulting from the processing | ± On par with competitors |
| ***System Requirements*** | BLE | BLE | BLE | ± On par with competitors |
| ***Platform*** | Win; Android | Win; Linux; Mac OS; Android | Win; Linux; Mac OS; Android; iOS | ± On par with competitors |
| ***Architecture*** | Atmel AVR ATXMega chipset | Atmel AVR ATXMega chipset | Atmel AVR ATXMega chipset | ± On par with competitors |
| ***Technology Readiness Level*** | 6 | 6 | 6 | + Opportunity to incorporate UCD inputs to create a distinctive product<br>- Competitors currently have TRL 9 |

**Maximilian**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Yes (C++) | C++ and Python bindings | | Has no dependencies |
| ***Possible extension points*** | Compatible with any C++ library. Developing all the time. | | | extensible - can be used in anything |
| ***Input data*** | Any data | | | unresctricted - can use any binary / non-binary form |
| ***Processing capabilities*** | - sample playback, recording and looping<br>- read from WAV and OGG files.<br>- a selection of oscillators and filters<br>- enveloping<br>- multichannel mixing for 1, 2, 4 and 8 channel setups<br>- controller mapping functions<br>- effects including delay, distortion, chorus, flanging<br>- granular synthesis, including time and pitch stretching<br>- atom synthesis<br>- realtime music information retrieval functions: spectrum analysis, spectral features, octave analysis, Bark scale analysis, and MFCCs | | | Fully featured and easier to compile when compared to other systems |

| | | | | |
|---|---|---|---|---|
| ***Output data*** | Events, memory arrays, audio | | | |
| ***System Requirements*** | windows or mac or linux or android or iOS or ARM embedded | | | |
| ***Platform*** | windows or mac or linux or android or iOS or ARM embedded | | | |
| ***Architecture*** | anything | | | |
| ***Technology Readiness Level*** | 9 | | | |

**Maven**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Yes (C++) | C++ and Python bindings | | Has no dependencies |
| ***Possible extension points*** | Compatible with any C++ library. Developing all the time. | | | extensible - can be used in anything |
| ***Input data*** | Any data | | | unresctricted - can use any binary / non-binary form |
| ***Processing capabilities*** | onset detection, beat detection, boundary detection (i.e detection of verse / chorus boundaries / instrument entries), novelty detection, interest measures, attention estimation, visualisation filtering | | | Fully featured and easier to compile when compared to other systems |
| ***Output data*** | Events, memory arrays, audio signals | | | |
| ***System Requirements*** | windows or mac or linux or android or iOS or ARM embedded | | | |
| ***Platform*** | windows or mac or linux or android or iOS or ARM embedded | | | |
| ***Architecture*** | anything | | | |
| ***Technology Readiness Level*** | 8 | | | |

**GestureAgents**

| | **Current State** | **1 Year** | **3 Years** | **Comparison with existing technologies** |
|---|---|---|---|---|
| **Programmability** | Yes (Python) | | | Decentralized approach |
| ***Possible extension points*** | Can be extended with gesture definitions, gesture recognizers, compatibility policies | | | recognizer agnostic (not forcing a single recognition strategy) |
| ***Input data*** | Any RT data | | | input-agnostic |
| ***Processing capabilities*** | event negotiation/assignation, gesture recognition | | | allowing arbitrary gesture recognition systems |
| ***Output data*** | gesture events | | | No rollback or ambiguity in events |
| ***System Requirements*** | Python interpreter / embeded in a larger program | | | Embeddable but not native |
| ***Platform*** | Multi-Platform | | | |
| ***Architecture*** | any | | | |
| ***Technology Readiness Level*** | 4 | | | |