Goldsmiths, University of London
Department of Computing

Interactive Machine Learning for User-Innovation Toolkits
An Action Design Research approach

José Francisco da Cruz Bernardo

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science of Goldsmiths, University of London

January 2020

Statement of Originality

I declare that the work presented in this thesis is my own unless otherwise stated. Every effort was made to make the contributions of others clear, with due reference to literature and acknowledgement of collaborative research and discussion.

Signed: Date:

Acknowledgements

The PhD journey has been for me the realisation of an old aspiration, and a challenging and highly rewarding inner journey. I was fortunate with the people and context that provided me with the opportunity, inspiration and necessary resistance to achieve such an intellectual growth. I would like to express my deepest gratitude to three people with whom I had the honour and pleasure to work with, and who have mentored and supported my research:

- Rebecca Fiebrink, my PhD advisor, who patiently supported my work and inspired me in so many different ways. Rebecca's critical thinking and communication skills, work ethics and attention to detail are but a few of the characteristics that make her so brilliant. Rebecca has been tireless at providing insight, encouragement, and helping me to systematically question my own thinking.
- Mick Grierson, my PhD advisor, for his warm leadership, practical wisdom and creative vision. Mick, who has been a driving force in academic research in creative technology and the creative industries, has strongly influenced how I think about use-inspired research in these areas.
- Atau Tanaka, who inspired me as a visionary artist and research leader, provided me first-hand with the opportunity to work with such a fantastic team and in an amazingly creative environment. Atau gently "kidnapped" me to Goldsmiths, motivated me to embrace the complexities of the RAPID-MIX project through design research, and ensured that I had the necessary resources to complete my research.

I would like to thank other amazing people which were either directly involved with this work, or with whom I had the pleasure of working, enjoying academic life, or simply enjoying and having fun with, in the last five years. They are:

- Marian Petre and Nick Bryan-Kinns, for their thorough examination, insightful discussion and comments, which helped me to significantly improve the quality of this work.
- The members of the extended RAPID-MIX team with whom I had the pleasure to collaborate and learn with during the past four years. That includes Fred Bevilacqua, Xavier Boisserie, Andres Bucci, Fabian Gilles-Renn, Sergi Jordà, Joseph Larralde, Sebastian Mealla, Panos Papiotis, Adam Parkinson, Alba Rosado, Hugo Silva, Jean-Baptiste Thiebaut and Michael Zbyszyński.
- Thor Magnusson, Chris Kiefer, Cécile Chevalier, Alice Eldridge, Halldór Úlfarsson and Paulo Gajanijo, at Sussex University, for the support, encouragement, and making the last stage of thesis writing more enjoyable and exciting.

- Other researchers, collaborators and friends from Goldsmiths and other institutions, including Alessandro Altavilla, Patricia Alessandrini, Jack Armitage, Hadeel Ayoub, Prashant Aparajeya, Baptiste Caramiaux, Terry Clark, Nuno Correia, Kate Devlin, Balandino di Donato, Marco Donnarumma, Bryan Dunphy, James Frink, Jerémie Garcia, Pierre-François Gerard, Myah Grierson, Rose Hepworth, Karen Hodgson, Steph Horak, Simon Katan, Matthew Yee-King, Pedro Kirk, Kevin Lewis, Konstantin Leonenko, Perla Maiolino, Louis McCallum, Kiona Neuhaus, Miguel Ortiz, Sam Parkinson-Wolfe, Evan Raskob, Blanca Regina, Esben Söerig, Nick Ward, Federico Visi, Caroline Zheng, and Robert Zimmer.

- Marco Gillies and Siân Prime, for their insightful and constructive feedback on my upgrade.

- Ivan Poupyrev, Nick Gilian, and Fumi Yamazaki at Google ATAP for the amazing experience and support at the Soli Alpha Developers program. Additional thanks to Nick Arner and Paul Batchelor for the teamwork with the *O Soli Mio* NIME'17 paper.

- Elizabeth Churchill, Molly Steenson and Philip van Allen, for the great experience and feedback at the AAAI SS-17 workshop *Designing the User Experience of Machine Learning Systems.*

- Luís Gustavo Martins, Sofia Lourenço, Ricardo Morais, Rui José, Helder Pinto, Isabel Ramos, Álvaro Barbosa, João Cordeiro, colleagues and mentors in Portuguese academic institutions where I was before coming to the United Kingdom, and with whom I also learned the ways of scientific research.

- Peter Challis, Augusta Quiney, Prosper and Arlo Challis, and Ripley, the super doggy, who were my extended family for the past four years and welcomed me into their home, in the lovely Telegraph Hill. Also Robin Leonard, Henna Shakoor and Nick Mooney, for the joyful times we enjoyed together there.

- My friends Bruno Lopes, Sandra Mega, Janete Pereira, Behzad Yazdani, Maria Costa, Francisco Calafate Faria, Gustavo Carneiro, Claudia Sousa Dias, who helped me to gracefully transition and settle in to London and keep the level of homesickness manageable.

- Rosa de Lourdes and Raymond Conte for all the care and amazing time spent in Paris for RAPID-MIX studies. To Paulo Magalhães and Ana Campos for the joyful break after a hard campaign, at beautiful Tavira.

- P. Cristiano Oliveira SJ, a man of faith and science who greatly inspired and influenced my current path.

- José Bernardo, Teresinha, Miguel, Avó Nocas, and all my family and friends whom I am very fortunate to have but are too many to name here.

- Maria Guilhermina Castro, for all the support and encouragement.

- Manuela Serra, my mother, who planted the seed, incited me to push forward, and always believed in me.

My research was supported by the EU Horizon 2020 Innovation Act H2020-ICT- 2014-1 Project ID: 644862, RAPID-MIX.

To Manuela, my mother.

Abstract

Machine learning offers great potential to developers and end users in the creative industries. However, to better support creative software developers' needs and empower them as machine learning users and innovators, the usability of and developer experience with machine learning tools must be considered and better understood. This thesis asks the following research questions: How can we apply a user-centred approach to the design of developer tools for rapid prototyping with Interactive Machine Learning? In what ways can we design better developer tools to accelerate and broaden innovation with machine learning?

This thesis presents a three-year longitudinal action research study that I undertook within a multi-institutional consortium leading the EU H2020 -funded Innovation Action RAPID-MIX. The scope of the research presented here was the application of a user-centred approach to the design and evaluation of developer tools for rapid prototyping and product development with machine learning. This thesis presents my work in collaboration with other members of RAPID-MIX, including design and deployment of a user-centred methodology for the project, interventions for gathering requirements with RAPID-MIX consortium stakeholders and end users, and prototyping, development and evaluation of a software development toolkit for interactive machine learning.

This thesis contributes with new understanding about the consequences and implications of a user-centred approach to the design and evaluation of developer tools for rapid prototyping of interactive machine learning systems. This includes 1) new understanding about the goals, needs, expectations, and challenges facing creative machine-learning non-expert developers and 2) an evaluation of the usability and design trade-offs of a toolkit for rapid prototyping with interactive machine learning. This thesis also contributes with 3) a methods framework of *User-Centred Design Actions* for harmonising User-Centred Design with Action Research and supporting the collaboration between action researchers and practitioners working in rapid innovation actions, and 4) recommendations for applying Action Research and User-Centred Design in similar contexts and scale.

Contents

List of Figures

List of Tables

First of all, there is the dream and the will to found a private kingdom [...] the sensation of power and independence loses nothing by the fact that both are largely illusions.
— Schumpeter (1934/1974, pg. 93)

Chapter 1

Introduction

This chapter begins by exposing the motivation and objectives of my research in Section 1.1. Section 1.2 presents a brief contextualisation of the research client project, RAPID-MIX, concerning its organisational structure, initial goals, the structure of research and development activities, as well as my role in the project. Section 1.3 introduces the research questions that my thesis pursues. Section 1.4 presents the main contributions of this thesis. Section 1.5 lists some of the publications that have emerged in the course of my research. Section 1.6 provides the outline of the remaining chapters of this thesis.

1.1 Motivation and objectives

Innovation studies indicate that research outcomes suffer from a systematic lag when traversing the innovation pipeline to become products, and often fail to make it to the market at all, e.g. (Mowery and Sampat, 1980). These limitations have also been explicitly identified in the field of Human-Computer Interaction, in which the design of new technologies with the adequate regard for human factors and context of use plays a fundamental role. Funding agencies and policymakers aiming to increase competitive advantage are currently tackling this problem and pushing for the acceleration of innovation and technology transfer, by bringing companies and research institutions to collaborate in joint innovation actions. Conversely, innovation studies have also been addressing patterns of user innovation that are emerging outside the firm and in the field of consumer products.

The recent developments in artificial intelligence research are generating a surge of interest around making machine learning more accessible to new groups of people—particularly people who are not ML experts—for problem solving and practical applications in various domains. Some academic research is focusing on the possibilities that artificial perception of the human body has to offer in terms of control and expressivity. They explore the users and contexts of use of systems for multimodal interaction, which embody rich sensing capabilities and the inference power of machine learning (ML) to leverage continuous and synchronous control and feedback. However, using ML is often difficult for those who are not ML experts or data scientists. As with other computer technologies that have seen mass adoption (e.g. email interfaces, word processing software, web technologies), broader adoption of ML can be facilitated by improving the usability of tools for employing ML. However, there are still many challenges to overcome in the path to the democratisation of these technologies.

Over the years, a number of general-purpose ML tools have been developed. These include both application programming interface (APIs) and graphical user interfaces (GUIs)—e.g. R packages (Kuhn et al., 2007), Weka (Hall et al., 2009), scikit-learn (Buitinck et al., 2013). With the recent breakthroughs in deep learning, we observe an intensive push of ML development toolkits and APIs into the hands of developers—e.g. Google Tensorflow (Abadi et al., 2015), Apple CoreML[1] and TuriCreate[2], Pytorch[3]. While most of these APIs target ML experts, some of them are catered to an audience of ML non-expert users. Many of these APIs remain difficult to use, however. As a recent software industry survey reports (Spain, 2017), factors such as the lack of apparent benefit, ML development experience and time to learn, seem to further affect the adoption of ML tools by developers.

Developers integrating ML into their applications are most likely to resort to infrastructural software developed by third parties (e.g. middleware, software libraries, online services, toolkits) as this will most likely provide more efficiency and cost effectiveness to the development process (Blackwell, 2002). ML infrastructural software should provide, at the minimum, the ability to train and evaluate existing ML algorithms on new data. Developers can integrate this functionality with their application code via API calls to extend their applications' capabilities. ML infrastructural software may also incorporate a certain degree of domain expertise; for instance, by making available complete ML pipelines with predefined choices of algorithms and parameters—e.g. Mellis et al. (2017)—pre-trained models for transfer learning (Jialin and Yang, 2010), or other functionalities that are likely to be useful for supporting application development in particular domains.

Designing a new ML API is a challenging task, however, and there are few empirical studies on proactive design approaches to accelerate and broaden innovation for creative technologies with this kind of developer tools. This thesis aims to fill this gap with a longitudinal action design research study within the innovation process of RAPID-MIX[4]—an EU Horizon 2020-funded Innovation Action—focused on the development of one of its main outcomes, the RAPID-MIX API. The design of RAPID-MIX API followed a user-centric approach where different stakeholders and prospective users—including academics, professional developers working in creative technology companies, non-professional end-user developers (Lieberman et al., 2006)—were engaged early and throughout the process.

The RAPID-MIX API is an ML toolkit designed specifically for software developers in the creative

[1] Core ML, `https://developer.apple.com/documentation/`, Accessed: 2019-09-15

[2] Turi Create, `https://github.com/apple/turicreate`, Accessed: 2019-09-15

[3] Pytorch, `https://pytorch.org/`, Accessed: 2019-09-15

[4] RAPID-MIX: Realtime Adaptive Prototyping for Industrial Design of Multimodal Interactive eXpressive technology

industries, such as creators of games and real-time music technologies. The RAPID-MIX API comprises ML libraries and learning resources that were designed and implemented in the context of RAPID-MIX project focused on the creative industries. The RAPID-MIX API also targets students, "hackers", and "makers" who may wish to develop new technologies using ML. The API aims to explicitly support interactive machine learning (IML) approaches (Section 2.1.2) to systems development, in which developers iteratively create, curate, and modify supervised ML training data sets in order to influence model behaviour. Such an approach is often useful in creative applications in which ML is used to design interactions with sensor (Hartmann et al., 2007), audio or video data (Fiebrink et al., 2011).

This work aims to expand the knowledge in the scientific community with an HCI perspective on innovation with interactive machine learning. The opportunity to do applied research as an insider in RAPID-MIX motivated the adoption of an Action Research methodology for this work. Action Design Research, an Action Research derivation, provides a participatory and iterative framework for generating design knowledge around the development of technological artefacts in organisational settings. This methodology appeared adequate, therefore, for examining the design of the RAPID-MIX API—within the organisational setting of the RAPID-MIX consortium—as an interactive machine learning toolkit for user innovation in the creative industries. The RAPID-MIX consortium, which included academic labs and small and medium enterprises (SMEs), led an iterative and user-centred approach to the design of ML developer tools with user communities such as hackers, makers, and music and creative technologists.

1.2 Project setting: Overview of the EU H2020 -funded Innovation Action RAPID-MIX

Real-time Adaptive Prototyping for Industrial Design of Multimodal Interactive and eXpressive technologies (RAPID-MIX) was an Innovation Action funded by the European Commission under the Horizon 2020 program (H2020-ICT-2014-1 Project ID 644862) that took place between February 2015 and March 2018. Joint innovation actions such as RAPID-MIX typically consist of multi-institutional organisations or *consortia*, where different members work collaboratively and share resources and assets to achieve common innovation goals. This type of initiative is focused on closer-to-the-market activities and aims at producing new or improved products or services (European Commission, 2014).

1.2.1 Consortium members

The RAPID-MIX consortium comprised three European research labs and five SMEs collaborating in the development of creative technology tools for rapid prototyping and product development. The RAPID-MIX consortium members were:

- Music Technology Group – Universitat Pompeu Fabra (MTG/UPF)
  —Music information retrieval, on-line communities and multimodal repositories;
- Embodied Audiovisual Interaction Group (EAVI) – Goldsmiths, University of London (GS)
  —Interactive machine learning, embodied, biosignal and audiovisual interaction;
- Institut de Recherche et de Coordination Acoustique Musique (IRCAM)
  —Interactive machine learning, sensing and audio technologies;
- PLUX—Wireless Biosignals, Physiological computing technologies;
- Reactable Systems—Tangible and mobile music applications;
- ROLI/JUCE—Cross-platform audio development platform and community;
- Orbe—Collaborative audio installations;
- AudioGaming[5]—Interactive audio technologies for sound designers.

The members of the RAPID-MIX consortium have devoted years of research to the design and evaluation of embodied and wearable human-computer interfaces for creative and music technology. They have developed and accumulated a significant portfolio of technologies for multimodal and expressive interaction, including hardware, software tools for interactive machine learning, digital signal processing, and cloud-based services and repositories for storing and visualising audio and multimodal data. Figure 1.1 depicts the members of the RAPID-MIX consortium and their respective technologies. The following section describes in more detail the specific technological advances that each member of the RAPID-MIX consortium contributed to the joint innovation portfolio.

1.2.2 Background technologies from the RAPID-MIX innovation portfolio

The joint innovation portfolio of background technologies that the RAPID-MIX project started off with included music information retrieval and digital signal processing software libraries, cloud-based repositories for storing and visualising audiovisual and multimodal data, and software libraries and high-level interfaces for applying interactive machine learning:

- BITalino (Guerreiro et al., 2013)—a low-cost toolkit to learn and prototype applications using body signals. It's for students, teachers, makers, artists, researchers, corporate R&D.
- CodeCircle (Fiala et al., 2016)—a collaborative live coding web platform developed for beginning coders and computing students. CodeCircle aims to support efficient experimentation and prototyping activities.

[5] AudioGaming, a French SME, had its official entry to RAPID-MIX in August 2015, replacing *Somethin' Else* Sound Directions ltd., a British SME which had started in the original line-up.

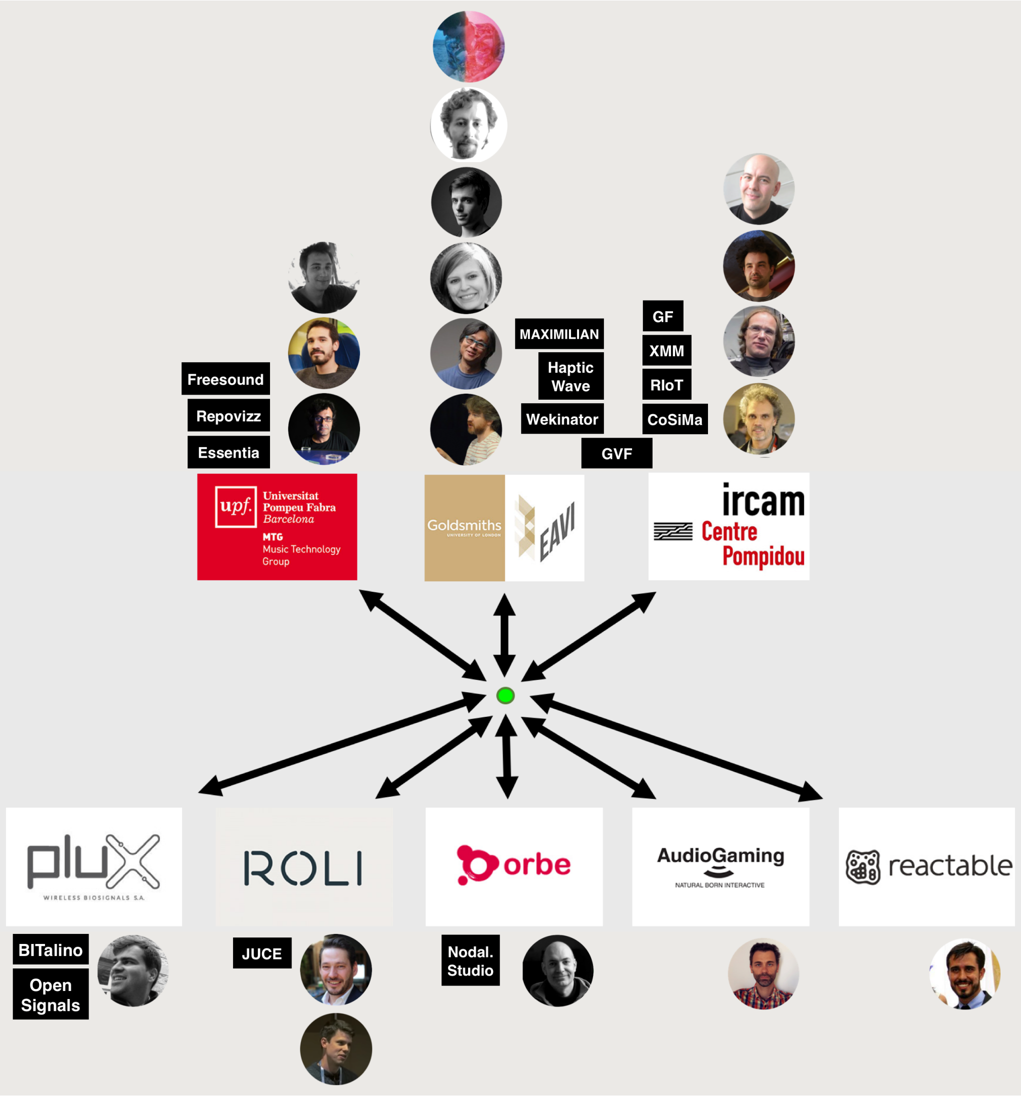


Figure 1.1: Map of RAPID-MIX consortium institutions, team members and technologies

- Maximilian (Grierson, 2010)—a C++ library, designed to ease the use of a wide array of audio features such as synthesis, filtering, FFT, etc. It has a syntax and program structure based on the popular 'Processing' environment.

- Wekinator (Fiebrink et al., 2011)—a general-purpose, standalone application for applying interactive machine learning. It provides a high-level interface to supervised learning algorithms and their parameters, and enables users to create and edit datasets rapidly, train and run models in a fast real-time workflow.

- HapticWave (Parkinson et al., 2015; Tanaka and Parkinson, 2016)—a haptic audio waveform display device that has been developed in collaboration with a group of audio engineers and producers with visual impairments for use in real-world recording studio environments.

- Gesture Follower (GF) (Bevilacqua et al., 2010)—a real-time gesture analyser that outputs "continuously" estimates of gesture parameters based on recorded templates; it is suited for selecting and synchronising visual or sound control processes to gestures.

- XMM (Françoise et al., 2013)—libraries for using Hierarchical Hidden Markov Models for classification and regression to model gesture and sound parameters, and for creating mappings between gesture and sound in interactive music systems.

- Collaborative Situated Media (CoSiMa) (Schnell, 2014)—a framework that provides components for audio processing, motion analysis and collective interaction based on web/mobile technologies, including HTML5, Javascript and Web Audio API.

- Gesture Variation Follower (Caramiaux et al., 2014)—a library for real-time gesture recognition and analysis that employs a template-based method using Sequential Monte Carlo inference.

- Interactive Audio Engine (IAE) (Schnell et al., 2009)—a synthesis engine in C++ for content-based audio processing. The engine extracts audio descriptors from recorded audio materials and provides asynchronous/synchronous granular synthesis and additive synthesis.

- Freesound.org (Font et al., 2013)—an online collaborative sound database where people from different disciplines share recorded sound clips under Creative Commons licenses.

- RepoVizz (Mayor et al., 2013)—a cloud service for collaborative data-driven research projects on performance and body motion, supporting structural formatting, remote storage, browsing, exchange, annotation, and visualisation of synchronous multimodal and time-aligned data.

- JUCE[6]—a wide-ranging C++ class library for building rich cross-platform audio applications and plugins for all the major operating systems.

Figure 1.1 shows how different RAPID-MIX technologies' intellectual property (IP) ownership mapped to RAPID-MIX institutional members. The figure depicts RAPID-MIX technologies in black boxes associated to the institutional logos of the IP-owning RAPID-MIX institutional partners. For instance, GVF, GF, R-Iot and CoSiMa were background technologies provided by IRCAM. However, GVF was a byproduct of a previous collaboration between IRCAM and Goldsmiths, and its IP was shared between both institutions. Some of the RAPID-MIX SME institutional members also acted as RAPID-MIX internal technology providers, e.g. PLUX and ROLI. Other RAPID-MIX SME institutional members such as Reactable and AudioGaming were mostly RAPID-MIX internal technology recipients.

1.2.3 Project goals and work plan structure

The RAPID-MIX consortium aimed at integrating their joint portfolio of technologies into one unified toolkit, the RAPID-MIX API. The design of this toolkit intended to break down the barriers and complexity associated with the use of sensor data for interactive and expressive applications,

[6] ROLI acquired JUCE in November 2014, https://forum.juce.com/t/juce-has-been-acquired-by-roli/13891

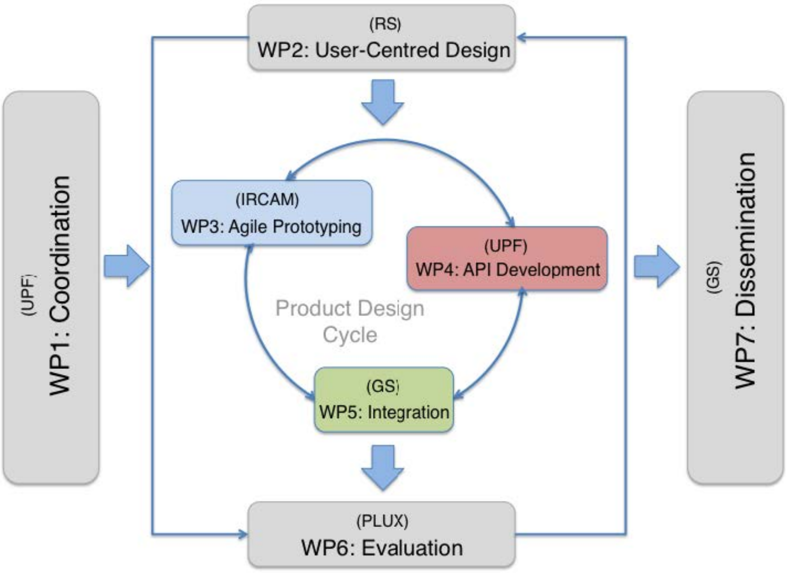


Figure 1.2: Structure of work packages in RAPID-MIX (Jordà et al., 2014, p.36)

by providing users with new accessible tools for digital signal processing and machine learning. The toolkit was intended to be comprehensive, easy to use and bring the different and necessary software components to integrate a wide range of novel sensor technologies into prototypes and products. The motivation for the design of this toolkit was, therefore, to empower development and innovation with upcoming products in the creative industries. The toolkit would enable rapid prototyping and fast product design cycles to reduce the lag that typically exists in technology transfer processes between laboratory-based and academic research, for instance, as well as in everyday commercial and creative applications.

To achieve these goals, the RAPID-MIX consortium planned upfront, at the stage of grant writing, to apply User-Centred Design (UCD) in the RAPID-MIX design process. This design process (Figure 1.2) consisted of an integrated iteration loop of concurrent prototyping, agile development of the RAPID-MIX API (Figure 1.3) and integration with “MIX products” (i.e. SME products for which the integration with the RAPID-MIX API would be outcomes of RAPID-MIX). The UCD process that had been planned consisted of another iterative loop where users would be engaged in order to inform the product design cycle and the general evaluation of its outcomes. Additionally, the RAPID-MIX consortium intended to share the learning about the process of bringing varied users directly into the development, at the scale of RAPID-MIX. The initial set of targeted users included the developers and designers of the RAPID-MIX SMEs in creative and music technology industries. Another set of potential users included individuals and companies outside the project, such as academic researchers, industrial designers, developers, musicians, hackers, makers, game developers and more.

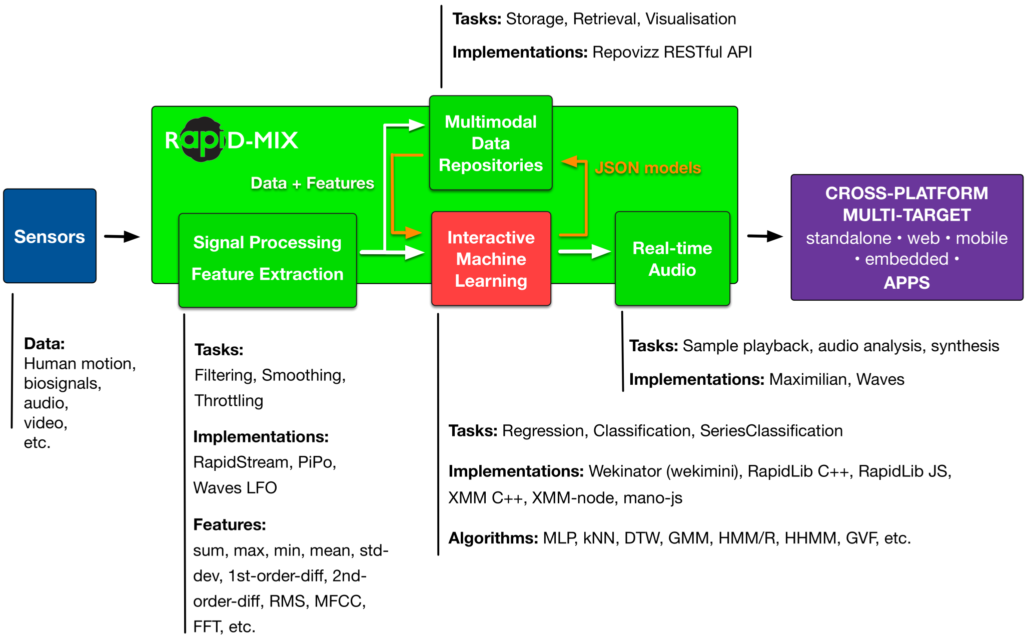

Tasks: Storage, Retrieval, Visualisation
Implementations: Repovizz RESTful API
RAPID-MIX
Data + Features
Multimodal Data Repositories
JSON models
Sensors
Signal Processing
Feature Extraction
Interactive Machine Learning
Real-time Audio
CROSS-PLATFORM MULTI-TARGET standalone • web • mobile • embedded • APPS
Data: Human motion, biosignals, audio, video, etc.
Tasks: Filtering, Smoothing, Throttling
Implementations: RapidStream, PiPo, Waves LFO
Features: sum, max, min, mean, std-dev, 1st-order-diff, 2nd-order-diff, RMS, MFCC, FFT, etc.
Tasks: Sample playback, audio analysis, synthesis
Implementations: Maximilian, Waves
Tasks: Regression, Classification, SeriesClassification
Implementations: Wekinator (wekimini), RapidLib C++, RapidLib JS, XMM C++, XMM-node, mano-js
Algorithms: MLP, kNN, DTW, GMM, HMM/R, HHMM, GVF, etc.

Figure 1.3: Diagram of the general structure of RAPID-MIX API

1.2.4 My role and responsibilities within RAPID-MIX

Since the RAPID-MIX project inception at the kick-off meeting at UPF, Barcelona, on February 2nd, 2015, I became an active member of the RAPID-MIX consortium, taking the role of action researcher. The kick-off meeting was my entry point into the research client system and where the Research/Client Agreement (one of the adopted research methodology principles which I will refer to, in Sections 3.2 and 7.4.1.1) took place informally. There, I presented the fundamentals of Action Research to the representatives of the different institutions participating in the consortium: the members of the coordination team from MTG/UPF, the leading researchers and team members of IRCAM and Goldsmiths, and representatives and team members from the SMEs (Figure 1.4).

As described in Section 1.2, the RAPID-MIX consortium—the client of my action research—was engaging in the challenge of a knowledge and technology transfer process, through the integration of a joint innovation portfolio. This task was complex, as it involved applying the User-Centred Design in the development of a technological artefact in the organisational context of RAPID-MIX. My presentation at the RAPID-MIX kick-off meeting highlighted the compatibility and complementarity of Action Research with the UCD methodology for the requirements of work package 2 (WP2 – *User-Centred Design*, figure 1.2) which had been initially proposed in the bid to the funding body, the European Commission.

Throughout the RAPID-MIX project, my role had a particular focus on WP2. In collaboration with other consortium members, I designed the *User-Centred Design Actions* framework (section 4.2), for data collection through interventions along the project, and planned, designed and deployed different interventions. In these interventions, I collected and analysed data and reported

Figure 1.4: Members of the RAPID-MIX consortium in the kick-off meeting day

in official deliverables to the European Commission. Specifically, I led the production of two deliverables—*D2.1 UCD Methodology* (Bernardo et al., 2015) and *D2.5 Final Design Specification* (Bernardo et al., 2018)—and collaborated in the remaining deliverables of WP2. Additionally, I collaborated in other RAPID-MIX work packages and deliverables, where my contributions included:

a) WP3 – *Agile Prototyping*—developing proofs-of-concept and software prototypes using RAPID-MIX technologies (Section 5.6.3);

b) WP4 – *API Development*—contributing to the design and development of the RAPID-MIX API (Section 5.6.1);

c) WP6 – *Evaluation*—designing and conducting user-centred evaluation studies (Sections 4.3, 4.4, 5.7 and 5.8);

d) WP7 – *Dissemination and exploitation*—designing communication and promotional materials, such as institutional websites, social media outlets and online community; organising, attending promotional events and representing the project publicly.

1.3 Research questions

The main questions that this thesis asks were triggered by a practical problem in the organisational context of RAPID-MIX. This problem will be examined in more detail in Chapter 5.2. These questions examine the process by which to inform the design of an artefact:

> *How can we apply a user-centred approach to the design of developer tools for rapid prototyping with Interactive Machine Learning?*
>
> *In what ways can we design better developer tools for accelerating and broadening innovation with Machine Learning?*

I broke down these research questions into the following set of questions:

1. Who are the users and what are their needs, goals and values?
2. What expectations do creative developers have and what challenges do they face when building Interactive Machine Learning systems?
3. How can we evaluate developer tools for rapid prototyping with Interactive Machine Learning and what can we learn from selected evaluation techniques?
4. What kinds of evaluation activities can we use to uncover problems of usability, usefulness and satisfaction of a toolkit for Interactive Machine Learning?
5. What are design trade-offs of developer tools for rapid prototyping with Interactive Machine Learning?
6. Can Interactive Machine Learning implement a good solution to the principles of User Innovation theory (von Hippel, 2005)?
7. How can the principles of User Innovation theory (von Hippel, 2005) inform the design of developer tools for rapid prototyping with interactive machine learning?

1.4 Summary of contributions

My primary contributions, which span empirical, methodological, and theoretical, include:

a) New understanding about the consequences and implications of applying a user-centred approach to the design and evaluation of developer tools for rapid prototyping of interactive machine learning systems. Specifically, this contribution includes:

   i) New understanding about the goals, needs, expectations, and challenges of creative ML-non-expert developers using tools for rapid prototyping of interactive machine learning systems.

   ii) Evaluation of usability, utility, and design trade-offs of a toolkit for rapid prototyping with interactive machine learning.

b) A new methodological tool for user-centred research and intervention, specifically:

   i) A new methods framework of User-Centred Design Actions which harmonises User-Centred Design with Action Research to support the collaboration of design practitioners and action researchers working in rapid innovation actions.

   ii) Recommendations for employing User-Centred Design in large-scale multi-institutional innovation actions, based on deployments of the User-Centred Design Actions framework.

c) Principles for the design of interactive machine learning for user innovation toolkits, derived from the application of User Innovation theory (von Hippel, 2005) and design spaces from Human-Centred Machine Learning (Amershi, 2012; Dudley and Kristensson, 2018) as working theoretical bases for informing design and analysis of interventions and artefacts for applying interactive machine learning.

Together, these contributions provide knowledge for informing the design of developer toolkits for accelerating and broadening innovation with interactive machine learning.

1.5 Publications

Journal Papers

Bernardo, F., Grierson, M., Fiebrink, R. (2018) User-Centred Design Actions for Lightweight Evaluation of an Interactive Machine Learning Toolkit. Journal of Science and Technology of the Arts (CITARj), Volume 10, No. 2 – Special Issue eNTERFACE'17
DOI: http://dx.doi.org/10.7559/citarj.v10i2.509

Conference Papers

Bernardo, F., Zbyszyński, M., Fiebrink, R., Grierson, M. (2017) Interactive Machine Learning for End-User Innovation. Designing the User Experience of Machine Learning Systems. Proceedings of Association for the Advancement of Artificial Inteligence (AAAI) Spring Symposium, Stanford University, Palo Alto, CA

Bernardo, F., Arner, N., Batchelor, P. (2017). O Soli Mio: Exploring Millimeter Wave Radar for Musical Interaction. Proceedings of New Interfaces for Musical Expression 2017, Aalborg University, Copenhagen, Denmark

Parkinson, A., Zbyszyński, M., Bernardo, F. (2017). Demonstrating Interactive Machine Learning Tools for Rapid Prototyping of Gestural Instruments in the Browser. Proceedings of Web Audio Conference 2017, Queen Mary University of London, London, UK

Ward, N., Ortiz, M., Bernardo, F., Tanaka, A. (2016). Designing and measuring gesture using Laban Movement Analysis and Electromyogram. 3rd Workshop on Full-Body and Multisensory Experience — BodySenseUX (UbiComp'16), Heidelberg, Germany

Technical Reports

Bernardo, F., Tanaka, A., Fiebrink, R., Parkinson, A., Meala, S., & Bevilacqua, F. (2015). D2.1 User-Centred Design Methodology. Retrieved from https://rapidmix.goldsmithsdigital.com/wp-content/uploads/2016/02/
D2.1-User-Centred-Design-Methodology1.pdf

Bevilacqua, F., Bernardo, F., Mealla, S., & Fiebrink, R. (2015). D2.2 Design Guidelines for Prototyping. Retrieved from http://rapidmix.goldsmithsdigital.com/wp-content/uploads/2016/02/
D2.2DesignGuidelineforPrototyping.pdf

Tilmanne, J., d'Alessandro, N., Barborka, P., Bayansar, F., Bernardo, F., Fiebrink, R., Heloir, A., Hemery, E., Laraba, S., Moinet, A., Nunnari, F., Ravet, T., Reboursière, L., Sarasua, A., Tits, M., Tits, N., Zajéga, F. (2018). Prototyping a New Audio-Visual Instrument Based on Extraction of

High-Level Features on Full-Body Motion Technical Report, in Proceedings of eNTERFACE 2015 Workshop on Intelligent Interfaces, Mons, Belgium

1.6 Outline

Chapter 1 provides an introduction to the thesis with the motivation for the topic of research and research objectives. I contextualise RAPID-MIX as the research client project, concerning its practical goals, organisational and work plan structure, and my role in the project. I state my research questions and the main contributions of my research. I conclude with a list of publications that resulted from my work and the outline of the subsequent chapters.

Chapter 2 presents a literature review on Interactive Machine Learning as a Human-Centred approach to machine learning, and on the intersection of Human-Computer Interaction with Innovation studies. I also cover fundamental concepts in Machine Learning to acquaint the reader with the most relevant terminology.

Chapter 3 introduces the reader to the adopted research methodology and research strategy design. I review the Action Research literature, covering two of its derivations, Canonical Action Research and Action Design Research, which were most relevant to my studies. I present the rationale for my decisions and state my positionality as a researcher in the Action Research spectrum.

Chapter 4 examines the primary research activities, findings and insights of the first Action Research cycle. This chapter covers preliminary research activities including initial diagnosis, definition and production of the *User-Centred Design Actions* methods framework. It also examines a set of interventions rooted in this framework, with RAPID-MIX consortium stakeholders and potential end user of RAPID-MIX technologies.

In Chapter 5, I employ the principles of the Action Design Research methodology in the formulation of my research problem, motivated by a practical problem of the project setting. I frame the research opportunity and cast it as an instance of a problems class, and I also state my initial research questions and contributing theoretical bases. Finally, I present the prior technology advances within the consortium up to this stage.

Chapter 6 examines the primary research activities, findings and insights of the second Action Research cycle. This chapter focuses on the RAPID-MIX API as the design artefact and unit of analysis, concerning design activities and rationale, interventions and user-centred evaluation. I examine two studies for lightweight formative evaluation of different RAPID-MIX API subsets with different user groups, and one general usability study employing the Cognitive Dimensions questionnaire with users of RAPID-MIX API.

Chapter 7 revisits the first main research question to discuss the findings and insights originating from the user studies of the action research cycles. This chapter focuses the discussion on consequences and implications of a user-centred design approach to ML infrastructural software and developer tools, for understanding better the usability and the experience they provide to ML-non-expert developers.

Chapter 8 employs the principles of Action Design Research in the formalisation of the emergent learning. I discuss the main findings against the analysis framework built upon von Hippel's User Innovation (von Hippel, 2005) theory and design abstractions from Human-Centred Machine Learning. I generalise from the problem of designing machine learning developer tools for broadening and accelerating innovation with ML-non-expert developers with the RAPID-MIX API, as a solution that consists of infrastructural software for rapid prototyping interactive machine learning in user-innovation toolkits. I derive design principles and argue that these theoretical constructs can be a useful contribution for guiding research and understanding the design of ML developer toolkits in the context of rapid innovation projects.

In Chapter 9, I conclude by specifying the outcomes of my research and main contributions. I reflect on the limitations of my research and on the quality criteria of Action Research that I implemented to ensure the rigour and relevance of my research. Finally, I outline research directions in HCI, interactive machine learning, and research and innovation through design, which my work has suggested as fruitful to pursue further.

There is nothing so practical as a good theory

— Kurt Lewin (1951)

Chapter 2

Literature Review

2.1 Interactive Machine Learning: Beyond Machine Learning

2.1.1 Brief overview of Machine Learning

Machine Learning (ML) is a subfield of Artificial Intelligence (AI), and more generally, "a natural outgrowth of the intersection of Computer Science and Statistics" (Mitchell, 2006, p. 3). Machine learning is considered particularly useful in contexts where a software application is too complex to be algorithmically designed, and where it is required to self-adapt to the environment to which it is deployed (p. 3). Machine learning is considered one of the most successful areas of computer science, with its methods successfully deployed in a wide range of real-world applications and domains spanning artistic, business, financial, medical and scientific. Typical scenarios where machine learning has been applied include, for instance, email spam detection, stock market forecasting, social media and product recommendation systems, health risk factors identification and prediction of disease outbreaks, control theory and robotics, and in information retrieval and semantic analysis of media content.

Machine learning comprises a collection of statistical analysis methods or algorithms that infer a mathematical model from data. Typically, an ML algorithm runs over a collection of data, the training set, during the training or learning phase. In this phase, the ML algorithm learns or infers the specific form of the model, a mathematical function that takes inputs and generates outputs. In the test or evaluation phase, this model is expected to correctly categorise new examples that differ from the training set data, what is known as *generalisation* (Bishop, 2006). The vast body of machine learning knowledge that has developed over time can be organised around key concepts such as workflows, learning strategies, tasks, algorithmic implementations, and their theoretical underpinnings. In brief, the most established strategies of ML are supervised, unsupervised, and semi-supervised learning approaches:

a) In the supervised learning approach, the goal of the learning system is known; the training dataset comprises pairs of inputs associated to a corresponding desired output. In supervised learning, algorithms are applied to two main tasks:

   i) Classification tasks, where datasets are classified into two or more classes (discrete output variable), by assigning a label or a category to a given input;

   ii) Regression tasks, where values of continuous output variables are predicted based on a set of input variables.

b) In the unsupervised learning approach, the goal is unknown; the underlying structure or concepts are to be learned from input data. An example of a task in unsupervised learning is clustering, in which the goal of the algorithm is to discover similarities between the input examples, organising them in regions called clusters.

c) In semi-supervised learning, the goal is partially known and is refined by considering more unlabelled data. Here, the training dataset comprises both examples of pairs of inputs and corresponding outputs, and examples comprising only inputs.

Common to all of the above mentioned machine learning strategies, is an iterative process of human knowledge discovery and optimisation, which typically involves trial and error (Xin et al., 2018b). In this process, ML practitioners have to collect, wrangle and clean data, perform data pre-processing and feature engineering, select machine learning algorithms and tune their parameters, evaluate the outcomes of applying the model, and potentially employ post-processing operations. Additionally, machine learning practitioners applying a supervised learning or semi-supervised learning approach have to label the data. In the unsupervised learning approach, there is still human intervention in setting up parameters (e.g. the choice of distance function, density threshold or number of expected cluster groups).

Some current trends in state-of-the-art applied machine learning research focus on the acceleration of the ML developer workflows—e.g. through *one-shot execution* of workflows (Sparks et al., 2016), new management approaches for feature selection (Zhang et al., 2016), or through optimisation of the workflow end-to-end and across iterations by the automated analysis and reuse of previous results (Xin et al., 2018a). Other strands in applied machine learning research study the reduction, or even the removal of human intervention from machine learning processes—e.g. automatic machine learning (AutoML) (Kotthoff et al., 2016) and meta-learning (Yao et al., 2018).

2.1.1.1 "Folk knowledge" of Applied Machine Learning

Until recently, traditional learning resources for machine learning have been found in specialist textbooks (e.g. Bishop, 2006; Hastie et al., 2017; Mitchell, 1997; Russell and Norvig, 2009), academic tutorial papers and surveys (e.g. Burges, 1998; Kirkwood, 2002; Schapire, 1999), documentation from the ML tooling ecosystem, and academic courses. These resources can impose a steep learning curve, given the density of their content, which aims for knowledge depth and completeness, and often includes theoretical demonstrations and mathematical proofs.

However, other resources have been produced with contrasting assumptions, such as pragmatism and a focus on practical application. For instance, one early and highly-cited learning resource in machine learning is Kohavi and Provost's (1998) *Glossary of Terms*, which lists machine learning concepts and standard terminology along with their definitions. Worthy of notice is how the authors refer to their design of the definitions: "definitions are not designed to be completely general, but instead are aimed at the most common case" (Kohavi and Provost, 1998, p. 1).

In *A Few Useful Things to Know about Machine Learning*, a popular paper among the machine learning community, Pedro Domingos claimed that the "folk knowledge" of machine learning—i.e. the knowledge that is crucial to successfully develop machine learning applications—is not readily available in most machine learning resources, despite being relatively easy to communicate (Domingos, 2012). He claimed that this leads to lag and "less-than-ideal" outcomes in applied ML projects. His contributions lessen this problem with twelve principles that synthesize the "folk wisdom" of machine learning. He used classification as an instance of a well-matured machine learning task, to illustrate and support his specific claims, claiming that his list of principles generalises across the broader applied machine learning space. Some of these principles include, for instance:

1. "Learning = Representation + Evaluation + Optimization" (p. 78) – to help ML practitioners to navigate the space of ML algorithms, Domingos proposed a combination of representation (algorithm), evaluation (scoring functions to distinguish between good and bad classifiers) and optimisation (searching for the right classifier in the classifier space) to find the right algorithm for a given problem.

2. "It's Generalization that counts" (p. 79) – typically, in applied machine learning, having a learning model generalise beyond test examples is the ultimate goal and Domingos recommends a careful separation between training and test data, and the use of techniques such as cross-validation to maximise the generalisation power of a ML model.

3. "Data alone is not enough" (p. 79) – according to Domingos, knowing the assumptions that specific learning algorithms incorporate and which are independent of the data they have to generalise from, is one of the key criteria for choosing an algorithm that adequately expresses the information that is in the data, and that delivers the value of machine learning in "getting more for less".

4. "Overfitting has many faces" (p. 80) – Domingos addressed *overfitting* as one of the big problems in machine learning. He discusses ways to understand this problem (e.g. "decomposing generalisation error into bias and variance") and methods to counteract it, such as cross-validation and regularisation;

5. "Intuition fails in high dimensions" (p. 81)– Domingos addressed another significant problem

in machine learning, the *curse of dimensionality.* He discussed how our intuition, which is grounded in a three-dimensional world, fails in high-dimensional spaces.

6. "Feature engineering is the key" (p. 82) – For Domingos, feature engineering is both the most important factor for the success of a ML project and where most effort is invested. He considered that feature engineering is difficult because it is domain-specific, and requires technical knowledge, intuition, creativity and "black art". This is contrasted with the learning part of a machine learning process as the quickest, given that general-purpose learning algorithms can be used and are mostly well-mastered.

7. "More data beats a cleverer algorithm" (p. 83) – According to Domingos, given the right set of features, if results do not get the expected accuracy, the next step is to either design a better algorithm (which takes longer), or gather more data (which leads to success more quickly).

8. "Learn many models, not just one" (p. 83) – Domingos pointed to the necessity of being well versed in the application of different learning algorithms, given that the best learner varies between applications.

9. "Simplicity does not imply accuracy" (p. 84) – Domingos claimed that simplicity should be preferred in principle, and dismissed considerations that relate simplicity to accuracy by tapping on examples and counter-examples to Occam's razor (i.e. "given two classifiers with the training error, the simpler will likely have the lowest error").

Domingos' (2012) principles make explicit some of the main assumptions and goals that characterise applied machine learning, such as maximising generalisation accuracy, avoiding overfitting, the prevalence of data and feature engineering over learning algorithms, etc. Most interestingly, Domingos' synthesis offers an example of a pragmatic attempt to challenge ML practitioners' assumptions, which provides direction towards the successful application of machine learning. His discourse balances different levels of depth, the breadth of applied machine learning knowledge, and practical utility.

While Domingos' (2012) heuristics are meant to complement a more conventional learning approach, they provide a succinct and pragmatic resource for practitioners at different levels of expertise to learn about the application of machine learning. However, the heuristics build upon the assumptions of a technology-centred perspective and are mostly directed towards the applied ML community—i.e. at the level of ML practitioners and soon-to-be experts. There is no consideration of the needs of developers applying these principles and ML tools for specific problem solving, or of the required skill level and related background knowledge for applying then.

More recently, with the advent of deep learning and the push for democratisation of machine learning, there has been an explosion of learning resources, from massive open online courses (MOOCs), YouTube video tutorials, blog tutorials and interactive web-based tutorials, pedagogical artefacts in the ML tooling ecosystem, through to courses from large technological conglomerates such as Google and Microsoft. One example of a course developed by a technological conglomerate for internal consumption by its software engineers[1] is Google's *Machine Learning Crash Course with TensorFlow APIs*[2].

Google AI subsequently released this course to the public, and among its resources was a set of heuristics that promotes the successful development of 'great' ML products—*Rules of Machine Learning: Best Practices for ML Engineering*[3] (Zinkevich, 2018). I will refer to this resource as *Rules of ML* in the remainder of this section.

In contrast to Domingos's (2012) "folk knowledge" of ML, *Rules of ML* provided a more extensive set of forty-three heuristics for machine learning development, grouped into four main stages (Zinkevich, 2018). In the preliminary diagnosis phase "Before Machine Learning", potential adopters are cautioned to identify a real need for ML before using it in the first place—"Rule #1: Don't be afraid to launch a product without machine learning." (p. 4). A machine learning need can be understood as some business metrics and complex heuristics that may be replaced by a ML process—e.g. "Rule #3: Choose machine learning over a complex heuristic" (p. 5).

The last three stages of *Rules of ML* focused on different phases of the machine learning system development life cycle. For instance, in the second stage, "ML Phase I: Your First Pipeline" (p. 5), comprises twelve heuristics for the first phase of the machine learning system lifecycle. These heuristics address the engineering aspects of building a working ML infrastructure: a) building an end-to-end ML pipeline, b) identifying metrics of interest and turning them into features, with owners and documentation, c) embedding simple and interpretable ML models first or no models at all, d) setting up and deploying unit and system tests for performance monitoring, and e) identifying an objective, for instance a business metric, as the main target of optimisation.

The third stage of *Rules of ML*, "ML Phase II: Feature Engineering " (p. 11), assumed a working end-to-end infrastructure and included twenty-one heuristics that address ML aspects such as: a) model deployment and improvement, b) model performance and behaviour measurement, and c) feature engineering tasks, such as exploration, combination and transformation. The last six heuristics grouped under the stage "ML Phase III: Slowed Growth, Optimization Refinement, and Complex Models" (p. 19) address a lifecycle phase in which the machine learning system hits a plateau. These heuristics are acknowledgedly vague or "blue-sky" as they are highly dependent on the ML systems' context, and recommend for more sophisticated tuning based on specific product and business goals.

Mostly, *Rules of ML* focuses on ML development for high-end business cases and large software infrastructures, and targets highly-qualified software engineers. This last aspect is conveyed by their guiding principle: "do machine learning like the great engineer you are, not like the great

[1] https://thenextweb.com/artificial-intelligence/2018/03/05/you-can-take-googles-machine-learning-crash-course-for-free-now/

[2] https://developers.google.com/machine-learning

[3] https://developers.google.com/machine-learning/guides/rules-of-ml

machine learning expert you aren't." (Zinkevich, 2018, p. 3). This guiding principle is quite explicit at identifying highly-qualified software engineers as class of non-expert machine learning users.

Interestingly, only a very small subset of guidelines of the third stage is entitled as *Human Analysis of the System* and referred to as "not taught in any machine learning class: how to look at an existing model, and improve it [...] more of an art than a science [...] several anti-patterns that it helps to avoid" (Zinkevich, 2018, p. 14). For instance, one of the heuristics from this subset—"Rule #23: You are not a typical end user." (p. 14)—recommends testing production using crowd-sourcing labelled data, and user experience methodologies, such as usability testing and user personae.

Previously, Patel et al. (2008) examined the difficulties of software developers adopting ML tools and the obstacles to their accessibility. The authors' principal focus was mostly on integrated development environments for applying ML such as Weka (Hall et al., 2009) or RapidMiner[4]. They argue that these general purpose tools provide insufficient guidance to developers who are unfamiliar with the application of ML algorithms. Patel et al. (2008) also claim that other ML tools designed specifically for accessibility are mostly limited to a particular domain and hide complexity by limiting the available options—e.g. Crayons (Fails and Olsen, 2003), Exemplar (Hartmann et al., 2007). Patel et al. (2008) claim that accessibility design trade off flexibility and general purpose of application, which is problematic if the developers' problems do not conform to the design assumptions of the tool.

In order to inform the design of both general and accessible ML tools for developers, Patel et al. (2008) ran interviews with both statistical ML experts and HCI researchers experienced in statistical ML development. Subsequently, they evaluated graduate developers undertaking structured model-building tasks using the think-aloud protocol and identified three main themes in the difficulties that developers face with ML tools:

1. *Following an iterative and exploratory process* – according to Patel et al., the linear dependencies of the ML workflow (i.e. problem formulation, data collection and labelling, feature extraction, algorithm selection, experimentation and evaluation) confound inexperienced ML developers. The lack of knowledge of inexperienced ML developers makes them prone to employ the process linearly and to become overly focused on a specific subset of the modelling process. Instead, success in a ML workflow often requires engaging in nonlinear exploration of all the subsets of the workflow.

   Patel et al. also found that the way in which ML tools divide tasks, and how they provide limited or partial support for end-to-end workflows, causes information gaps that undermine exploratory and interactive development. For instance, tools which enable the application of ML techniques without the need to implement or understand underlying algorithms, prevent developers from being able to understand and inspect the state of the algorithms; this

[4] RapidMiner, Lightning Fast Data Science for Teams. `https://rapidminer.com/`

can contribute to spending excessive time on algorithm selection and underestimating other aspects of the ML workflow (e.g. data and feature selection). Also, tools that are agnostic to feature development and selection break the relationship of feature output with raw data, causing a barrier to debugging. Most ML tools also lack support for progress tracking and for experimental history record keeping (i.e. previously-used strategies including combinations of algorithms, features and achieved model accuracy), pushing developers to adopt fragile techniques for documenting the exploratory process.

2. *Understanding the relationships between data and model behaviour* – developers often exhibit the 'black box' mindset—they expect algorithms to be immediately effective, make few attempts at parameter configuration, engage in random experimentation without striving to understand why ML systems fail or how to debug them. ML learning tools with opaque implementations of algorithms reinforce this mindset and encourage bad experimentation practices, instead of supporting a more principled approach to problem solving and model improvement, based on understanding, exploration and visualisation of models and data.

3. *Evaluating the performance of models in the context of applications* – developers showed difficulties evaluating model accuracy (difficulties with collecting data using the IID assumption, with structuring training and test data sets) and at making models generalise to new data. They also showed difficulties at managing other concerns beyond model accuracy and traditional evaluation metrics (e.g. privacy trade-offs in data collection, computational cost of models and implications for responsiveness of interactive applications).

According to Patel et al. (2008), a successful ML tool would encourage and enable developers to try the right experiments and pursue effective strategies for implementing ML. Based on the their previous findings, they provide a set of design implications for how to build both general-purpose and accessible ML tools:

1. *Building statistical ML tools* – ML tools should support an exploratory and iterative approach to end-to-end workflows. This should be achieved through the integration of all subsets of the workflow, such as data collection and feature creation, and by counteracting any potential information gaps. This includes providing libraries of common feature computations for well-studied domains, logging of the entire experimentation process using lightweight metadata to preserve the origin of data, and ideally, mining this information to suggest to developers new experiments with different combinations of data, features and algorithms.

2. *Designing ML algorithms* – ML algorithms should be designed for understanding rather than for encouraging their use as black boxes. Developers need to understand and to interpret classifiers, and to understand how and why ML systems fail. This implies, for instance, providing support for visualisation: of data, features models, relationships between these entities, for examining misclassified data.

3. *Presenting statistical ML* – ML tools should support developers in attaining the goal of applying ML algorithms and techniques to make a system work. ML tools should ease the transition between the necessary exploration leading to model creation and the deployment of that model in a client application. ML tools should also use repositories with information about what combinations of algorithms, features and data work well.

2.1.1.2 Machine Learning tools in Creative Technology

Machine Learning has also been increasingly applied in more creative endeavours, such as the generation and control of media such as image, video and music. MnM (Bevilacqua et al., 2005) is a toolkit that allows users to create custom gesture-to-sound mappings using statistical methods such as principal components analysis[5], hidden Markov models[6] and other algorithms. The toolkit is implemented as a suite of Max/MSP externals (i.e. processing components that are used within Max/MSP's graphical patching environment).

The Gesture Recognition Toolkit (GRT) (Gillian and Paradiso, 2014) is an OSS, cross-platform C++ library designed to make real-time machine learning and gesture recognition more accessible for non-specialists. *ml.lib* (Bullock and Momeni, 2015) is an OSS machine learning toolkit designed for two domain-specific data flow programming environments, Max[7] and Pure Data[8]. *ml.lib* was implemented as a set of modules that wrap the GRT C++ library (Gillian and Paradiso, 2014) and execute within these environments as external components. The decision behind this choice was due to GRT's "wide range of implemented algorithms, simple design, straightforward C++ interface, pre- and post-processing functions and orientation towards artistic applications, specifically real-time gesture analysis." (Bullock and Momeni, 2015, p. 4). Other aspects included in the design rationale of *ml.lib* include:

- enabling users with minimal prior knowledge to experiment with and integrate a wide range ML techniques into their interactive arts projects.
- taking advantage of data-flow programming environment affordances, which include a) rapid prototyping and b) integration of different types of multimedia, c) high-level abstraction (i.e. hides away threading and memory management), and d) integrated documentation for externals with interactive examples.
- maximising learnability and discoverability through "a simple, logical and consistent, scalable interface".

[5] Principal component analysis (PCA) is an advanced statistical method that converts observations of potentially correlated variables into a set of linearly uncorrelated variables (the principal components). PCA is often applied for dimensionality reduction high-dimensional data sets (Turk, 2001).

[6] A hidden Markov model is a Markov chain for which the state is only partially observable. A Markov chain is a method for modelling complex systems using random processes and probability, sequences of possible events and interdependent states.

[7] Max/MSP, `https://cycling74.com/products/max/`

[8] Pure Data, `https://puredata.info/`

- providing portability and maintainability through the use of a cross-platform and multi-target technology stack that supports different desktop operating systems, such as Mac OS, GNU/Linux and Windows, and embedded hardware architectures and processors, such as Raspberry PI and ARM, respectively.

Fiebrink et al. (2011) explored the interactions of users with machine learning in the context of compositional and music practices. Wekinator, a general-purpose standalone application for applying machine learning, provides a high-level interface to supervised learning algorithms and their parameters, enabling users to rapidly create and edit datasets, and to train and run models in real time. It does so by enabling a play-along approach to performance using interactive machine learning by connecting sensors to end-user musical software.

2.1.2 A Human-centred perspective to Machine Learning

In the domain of Human-Computer Interaction (HCI), ML has been applied to supporting effective human interaction, for instance, inferring user intent in mixed-initiative and adaptive interfaces (Horvitz, 1999), modelling user activities in context-aware applications (Dey et al., 2004), developing new input modalities through analysis and interpretation of sensor data streams (Jaimes and Sebe, 2007).

Other HCI research efforts have focused on the interaction between humans and intelligent systems, and on the design of solutions, workflows and techniques that make ML easier and more accessible to a broader group of potential users. According to Talbot et al. (2009), with the increasingly widespread application of ML, the HCI community faces the challenge of providing adequate tools that allow for the democratisation of ML.

Among the existing approaches, Interactive Machine Learning (IML), which was first defined by Fails and Olsen (2003) as a new ML paradigm, features a workflow with rapid cycles of training and corrective feedback of classifiers with human supervision. Dudley and Kristensson (2018) provide a more recent and comprehensive definition:

> *Interactive Machine Learning is an interaction paradigm in which a user or user group iteratively builds and refines a mathematical model to describe a concept through iterative cycles of input and review. Model refinement is driven by user input that may come in many forms, such as providing indicative samples, describing indicative features, or otherwise selecting high-level model parameters. Interactive Machine Learning is distinct from classical machine learning in that human intelligence is applied through iterative teaching and model refinement in a relatively tight loop of "set-and-check." In other words, the user provides additional information to the system to update the model, and the change in the model is reviewed against the user's design objective. Under this workflow, the change in the model at each iteration may be relatively small.*

> *This is in contrast to more traditional machine learning approaches where the workflow requires wholesale pre-selection of training data and significant changes in the model per execution step.*
>
> (Dudley and Kristensson, 2018, p. 4)

Amershi et al. (2014) also described the IML workflow as more rapid, more focused and incremental when compared with classic machine learning (CML). They considered the CML process laborious and sometimes inefficient. For Amershi et al., users adopting the IML workflow interactively examine the impact and steer the behaviour of the classifiers through focused experimentation and low-cost trial and error. Figure 2.1 below shows two schematics that Amershi et al. used to illustrate the main differences between the IML and CML workflows.

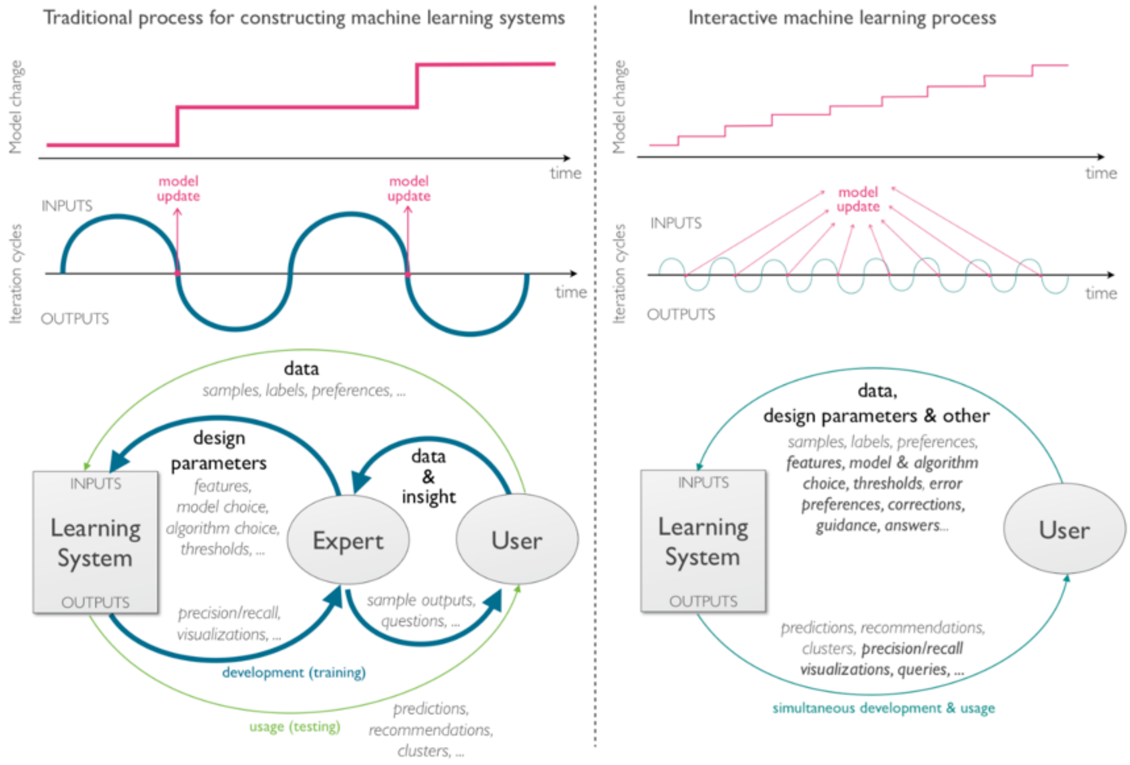

Traditional process for constructing machine learning systems
Model change
time
model update
model update
INPUTS
Iteration cycles
OUTPUTS
time
data
samples, labels, preferences, ...
design parameters
features, model choice, algorithm choice, thresholds, ...
data & insight
INPUTS
Learning System
OUTPUTS
Expert
User
precision/recall, visualizations, ...
sample outputs, questions, ...
development (training)
usage (testing)
predictions, recommendations, clusters, ...
Interactive machine learning process
Model change
time
model update
INPUTS
Iteration cycles
OUTPUTS
time
data, design parameters & other
samples, labels, preferences, features, model & algorithm choice, thresholds, error preferences, corrections, guidance, answers...
INPUTS
Learning System
OUTPUTS
User
predictions, recommendations, clusters, precision/recall visualizations, queries, ...
simultaneous development & usage

Figure 2.1: Comparison between CML and IML workflows (Amershi et al., 2014). Copyright © 2014, Association for the Advancement of Artificial Intelligence. All rights reserved.

Previously, Ware et al. (2001) also referred to IML as a process in which users engage in the autonomous generation of a classifier using simple GUIs. However, this approach might be considered fundamentally different from what is currently considered as IML. Their approach is based on a visual and interactive technique that enables users to visualise data via GUIs, and to generate classifiers which are inferred from decision boundaries manually drawn over data. This approach does not offer the possibility to select algorithms or set parameters, and it is actually restricted to one learning algorithm (i.e. decision trees classification algorithm). Interactive Machine Learning might be considered distinct from active learning, a human-in-the-loop pattern in which an algorithm prompts a human annotator to provide labels for sets of unlabelled examples and for cases where the algorithm has a low confidence (Settles, 2009).

In her doctoral thesis, Fiebrink (2011a) takes an HCI perspective on ML, focusing on how to make supervised learning algorithms more usable when applied to the domain of real-time interactive computer music systems. Fiebrink highlights a set of areas from this domain which have complex computational problems that can be addressed effectively using supervised learning—digital musical instruments design, semantic audio analysis and gesture recognition. Of particular relevance is the mapping problem in the realm of digital musical instruments (DMI) design. This an important example of a potentially very complex relationship between performer gestures and sonic response, which can be effectively modelled using supervised machine learning. In this relationship, the data from a variable array of sensors (e.g. inertial motion units, biosignals sensors, computer vision, etc.) is mapped to synthesizer parameters or to triggers for audio control processes using an IML approach supported by Wekinator.

Fiebrink et al. (2011) studied how musicians and students interact with Wekinator; students learned how to create datasets and to provide samples to train Wekinator; they learned how to map a gesture space into a sound space, and to interpolate between unobserved gesture positions. Students also learned to recognise noise in the samples and to refine the training data set to obtain a more effective classifier, iterating consistently through designs. Most interestingly, Fiebrink et al. found that users preferred direct model evaluation to more typical methods, such as cross-validation; and that musicians trained their musical instruments, and simultaneously got trained on the interactive nature of Wekinator.

Amershi et al. (2014) highlighted the importance of studying users of machine learning applications based on the tighter link that is established between learning systems and their users. Amershi found that end user's goals had a significant impact on the success of the features that were designed for and provided by the system. The considerations about whether end users wanted to train an accurate or reusable model, or whether they were interacting directly with machine learning, for instance, evidenced the need for different design requirements, such as providing a greater or lesser number of alternative mechanisms for system feedback and end-user control, different levels of guidance, and of complexity in explanations about the quality of the model. Amershi et al. (2014) studies also showed that users were empowered to create better models when comparing previous research techniques with newly introduced techniques such as overview-based example selection.

Yang et al. (2018) conducted empirical user studies with ML non-expert users in order to bridge the gap between accessible IML tools design and how users build models in real-life contexts. Yang et al conducted two studies: an empirical study with non-experts building ML models with the assistance of hired ML consultants, and a survey with non-experts about their experiences with ML. Based on their results, Yang et al. described the unique potentials of non-expert ML tools and the pitfalls that non-expert users are susceptible to. For instance, they found that ML non-experts' are motivated to build models without a clear goal in mind. They are driven by circumstances, such as having a dataset available, gaining insights about data and about the features choice, improving heuristics-based solutions, without problems that require a high level of accuracy. Yang et al. also found that non-experts are more satisfied and trusting about the learning results of ML model and can thus erroneously deploy invalid models to production.

2.1.2.1 Interactive Machine Learning tools

Talbot et al. (2009) presented EnsembleMatrix, an interactive visualisation system that allows users to explore the properties and the space of combinations of classifiers through comparison and contrast, to build combinations of classifiers based on ensemble techniques. EnsembleMatrix implements a view where confusion matrices are reordered to provide users with visual insights about the quality of the results of component classifiers. Additionally, by providing two operations, partitioning and linear combinations, users are able to create an ensemble classification system by discovering appropriate combination strategies. Talbot et al. showed that users were able to rapidly combine multiple classifiers and produce a very accurate ensemble classifier. Moreover, EnsembleMatrix also enabled the discovery of techniques that outperformed the state-of-the-art in computer vision at the time.

Kulesza et al. (2014) aim to improve the reliability and efficiency of one particular ML subset for amateur users—the labelling process. They introduced "structured labelling" tools to help people define and refine their concepts as they observe data in the labelling process. This technique addresses the problem of concept evolution in machine learning. Concept evolution refers to "the labeller's process of defining and refining a concept in their minds, and that can result in different labels being applied to similar items due to changes in the labeller's notion of the underlying concept" (p. 3075). The underlying concept refers to an abstract notion of the target class that a person is labelling for. The problem with concept evolution is that it leads to labelling inconsistency which is harmful for the machine learning process, despite more resilient algorithmic approaches to noise in the data. Structured labelling was found to help people to label more consistently in the presence of concept evolution, than with a traditional labelling technique.

Patel et al. (2010) introduced Gestalt as a general-purpose programming environment for lowering the ent barrier of applying machine learning. Gestalt provides an integrated environment with support for implementing a classification pipeline and for analysing the data as it moves through the pipeline. Gestalt provides a structured workflow that is common for many classification tasks; the pipeline supports problem representation, as domain-specific tools do, while maintaining the flexibility of a general-purpose tool by affording scripting for each pipeline stage. Gestalt also dismisses the need for management of data conversion or for using external tools for visualisation. It provides different kinds of views over data and allows developers to rapidly customise data visualisations at different stages of the pipeline; it also provides connected visualisations for comparing raw data, attributes and classification results, all in one view. Patel et al. demonstrated empirically that these features actually help developers succeed at applying machine learning.

Amershi et al. (2015) present ModelTracker, an interactive visualization that allows for performance analysis during model building and also enables error inspection and debugging. ModelTracker aggregates traditional summary statistics and graphs, for conveying model performance, with performance metrics in the same visualization, very much in the style of Information Visualization. ModelTracker "encourages informed approach to model building in machine learning" (p. 338) by preventing the cognitive switch between the task of building a model and the task of debugging model performance and analysing prediction errors. Amershi et al. conducted an experiment for six months with ML practitioners that used ModelTracker for building real models for research and product deployment. The usage analysis of the six-month logs of the study demonstrated that ModelTracker was adopted over some of the current tools for performance analysis and debugging which indicates that its integrated approach with information visualisation is more effective.

More recently, Françoise et al. (2016) discuss the potential of interactive visualisations for shifting "the perception of machine learning models from black-box processes to transparent artefacts that can be experienced and crafted" by exposing "the behaviour and internal values of machine learning models rather than their sole results" (p. 1). They argue in favour of the development of new visualisation techniques that extend beyond the traditional visual representations of the process (training data, cross-validation and inference results) to cover all the steps of the process. They point out the "creative potential of interactive visualisation that unveil model's affordances" (p. 5) and the pedagogical value of feedback in building the user's mental model. They introduce Gaussbox, a proof-of-concept that provides interactive visualisations and which allows direct manipulation of Hidden Markov Models (HMMs). In Gaussbox, internal elements and parameters of an HMM, including Markov chains, covariance matrix and mean of the Gaussian, are mapped to interactive and responsive visual representations in the graphical user interface. There are no user studies to validate the tool but Gaussbox raises interesting questions about the affordances of models, attributes, boundaries and other internal elements used in ML for meaningful and direct manipulation.

2.1.2.2 Design spaces of Human-Centred Machine Learning

Based on previous research and on insights from her own experiences in the design and development of ML systems, Amershi (2012) distilled the factors that impact and constrain how end users interact with machine learning systems. She abstracted this knowledge into a descriptive design space for end-user interaction with ML, which she proposes as a foundation and reference for research and development of IML systems. Amershi intended to provide a common language for understanding and discussing IML design, and this design space provides a framework for the identification of new design challenges, opportunities, and to support systematic approaches to systems design and analysis.

The design space for end-user interaction with machine learning (Amershi, 2012) comprises two main groups, Design factors and Design dimensions. The design factors are organised and characterised according to the kind of impact they have in IML design. Amershi identifies as main factors the end-user goals and associated contexts of use of machine learning, along with the design constraints that emerge from these. End-user goals and contexts refer to the intended outcomes and underlying reasons for using machine learning, and to the specific contexts in which machine learning will be used. They are briefly described as follows:

- *Intended Product* – the desired outcome of the IML process; this could be either a ML model, or the outcomes of applying the model—i.e. the predictions or classification only. Related to this is the concept of model reusability, where a model can be trained for reuse in future tasks, or contrastingly, trained for a transient task and discarded after providing the end user's desired output. As Amershi suggests, this factor has implications to the time and effort the end user is willing to commit to improve the system's understanding and generalisation performance. The Intended Product can also impact the level of trust the user has in the system for tasks that entail variable risk for health or integrity.

- *Interaction Focus* – whether the end user's focus will be placed on the IML loop or rather on a domain specific goal. If the focus is on the IML process, this factor impacts system design by means of providing a rich interface with functionalities that support the end user to interact effectively with the machine learning process (e.g. explanations, confidence metrics, impact counts, etc.). If the focus is on the domain task, the IML loop takes more of a background role and is used mainly as a means to accomplish the specific goal. In such case, the user might not even be aware they are interacting with a machine learning process.

- *Evolutionary Needs* – refers to a factor that exists along a spectrum ranging from fixed models to models that need to evolve according to changing user needs or scenarios of use. Fixed models are stable, as they capture concepts that are objective and stable themselves. Evolving models change through time to capture changing needs that result from the emergence of new data or more subjective concepts. Amershi suggests that this factor may impact the frequency, timing and mode of model updates.

- *Concept Flexibility* – refers to whether the concept to be acquired by the model is objective and well defined according to the user's perspective, or whether it is flexible or open-ended and there is more of an exploratory purpose behind the process. This factor affects the type of feedback that the system should display regarding diversity of results and predictability in terms of matching outcomes.

- *Performance Requirements* – such as accuracy, efficiency and precision are typically associated with the context of use. They also relate to the compromise that may be accepted or required regarding specific standards or performance levels in terms of admissible error rates and confidence that a model should achieve. This factor will impact the time and effort spent on training and optimizing the model, and the design of mechanisms used to provide feedback about performance metrics.

- *Model Ownership* – refers to the issue of whether a model is intended for individual or shared use. Amershi expects this factor to become more relevant in the future given that integrating pre-defined models might be more useful to some users instead of building them from scratch. This factor might impact the effort in assessing the model's generalisation performance, in providing descriptions for sharing them, and in adapting them to the new users.

Amershi (2012) also characterises design constraints as factors that are associated with end user's goals and contexts, with respect to three main aspects:

- *Data Constraints* – refers to factors relating to the nature, the source and the volume of data that shapes the IML process. The nature of data is related to the adequacy of features and transformations that may have to be applied in order to provide good examples to the learning algorithms. The source of data is related to aspects around the diversity of the origin of data, such as static data, data streams, intermittent or multiple sources. The volume of data refers to the quantity of data that the user is required to process. Amershi suggests that these factors may impact the design of the system so that it accommodates the specifics of the data source, provides sampling, pre-processing and feature transformation routines. It may also impact the timing and frequency required for the user to interact with the IML process, and to have evaluation mechanisms of generalisation performance available.
- *End-user Constraints* – refer to factors that are intrinsic to the end users, such as their machine learning expertise and mental model of the IML loop, the familiarity with the specificities of the domain and data that will be used in the process, the number of people that will engage in the machine learning process, their motivation and any other individual characteristics that may impact the process. Amershi suggests that these factors may impact system design in many ways, such as the level of guidance that the system should provide, the necessary information, including how it is presented to maximise the user's understanding, how engaging the interface might be, and how to support the coordinated effort of multiple users in the same process.
- *Environmental Constraints* – refer to the display size and surroundings, which may impact the amount and quality of feedback provided by the system, and the conditions in which this feedback is delivered to the user. Feedback must be designed with consideration to different screen sizes (if there is a screen at all) and for interruptions that may interfere with the end user's attention.

Amershi (2012) structures design dimensions into a group of categories that outline how end-user interaction techniques with machine learning were designed in previous research.

- *System Feedback* – refers to how the machine learning system communicates its understanding to the end user. Amershi identifies several attributes that characterise system feedback more intrinsically, in terms of *Type* (the main elements of the machine learning process that the system must present feedback on, such as input data, predictions, features, internal logic representation and processing state), *Granularity* (the level of feedback detail) and *Variety* (multiplicity of alternative mechanisms over the same feedback type). Other attributes characterise mechanisms for system feedback in terms of how and when feedback is exerted, such as *Push versus Pull Feedback* (system feedback prompts versus end user requested) and *Update Visibility* (making explicit the impact of new data in the IML process).

- *End-user Control* – refers to the mechanisms by which the end user can exert control and influence the machine learning system's understanding and behaviour. As one can observe in Figure 2 above, there a significant correspondence between dimensions of feedback and control, which often implies an association between their dimension, but that is not always the case. Amershi identifies several dimensions that characterise control in terms of *Type* (the main elements of the machine learning process which allow control by the user; for instance, creation, selection and labelling of examples, manual editing of data, selecting and filtering features, creating and manipulating classes, configuring algorithm parameters, etc.), *Granularity* (the level of control over the control types) and *Variety* (multiplicity of alternative user controls that are presented). With *Explicit versus Implicit Control*, Amershi refers to whether the system allows direct guidance over the different stages of the learning process, or if it is happening passively, potentially without user awareness. *Mandatory versus Elective Control* refers to elements of control that are either required from the user or optional. Finally, *Guidance* refers to the amount of direction in steering the system provided to the user (including system suggestions, examples, queries or interface scaffolding that restricts or influences the user interaction).

- *Temporal* – refers to the time dimension, such as timing and frequency of interaction of user with a machine learning system. Amershi states this design dimension relates to how the model update frequency is affected by direct user control (Timing of End-User Control), by system control (Timing of System Updates), or for how long the user feedback affects the model (System Memory), or if there is support for backwards (undo operations) and forward control (adding more labels)(Control Direction), or if the system supports either reflective or predictive feedback (Feedback direction) and the progressive evolution of the end user's mental model (Human Learning).

Another related contribution is the design space formulated more recently by Dudley and Kristensson (2018). Dudley and Kristensson frame IML as an HCI task, for which they argue that user interface (UI) design is critical. They used a systematic review, as Amershi did, but more specific to IML, which included workflows, design principles, and domain-specific IML system implementations. Dudley and Kristensson formalised the design space for comprehensive IML, by breaking down a generic and 'complete' IML system into its structural and behavioural components. Dudley and Kristensson's design space focuses on the critical task of interface design of an IML system and how it should support the different interactive model-building tasks. This design space breaks down IML systems structurally into a four-component architecture. This architecture includes the elements User, Model, Data, and User Interface (UI). The UI element is further divided into four sub-elements that should be part of the IML system's UI, Sample Review, Feedback Assignment, Model Inspection and Task Overview. This design space breaks down the interactive model-building task into six sub-tasks: Feature Selection, Model Selection, Model Steering, Quality Assessment, Termination Assessment and Transfer.

I consider these design spaces an important contribution which I cover more in depth through this thesis and will employ in the problem formulation (Chapter 5.2) and formalisation of learning (Chapter 7).

2.2 Where Human-Computer Interaction meets Innovation Studies

Innovation has become an important and recurrent term in today's society. Edison and Torkar (2013), in the context of a comprehensive survey on innovation measurement practices in the software industry, elected Crossan and Apaydin (2010) definition out of 40 definitions of innovation "for its comprehensive coverage of the identified aspects of innovation as found in literature" (p. 1401). I adopt this as my operational definition of innovation since it is fairly recent and originates from the context of software development:

> *Innovation is: production or adoption, assimilation, and exploitation of a value-added novelty in economic and social spheres; renewal and enlargement of products, services, and markets; development of new methods of production; and establishment of new management systems. It is both a process and an outcome.*

(Crossan and Apaydin, 2010, p. 1155)

Recent studies in the United Kingdom show that innovation is a significant driver of business growth, with innovative firms growing two to five times faster than firms that do not innovate (Nesta, 2009). There is also evidence of an increasing level of user involvement in innovative activities in the United Kingdom (von Hippel et al., 2010). The importance of innovation can also be observed in the public policy domain, such as in the OECD Innovation Strategy 2015 agenda (OECD, 2015), which defines and prioritises recommendations for public policies on innovation, more specifically to:

1. *Strengthen investment in innovation and foster business dynamism.*
2. *Invest in and shape an efficient system of knowledge creation and diffusion.*
3. *Seize the benefits of the digital economy.*
4. *Foster talent and skills and optimise their use.*

5. *Improve the governance and implementation of policies for innovation.*

(OECD, 2015, p. 2)

Innovation research at the meso level focuses on systems of innovation as the subject of study, and at a socio-economical perspective, on the interplay between institutions, such as universities, firms and funding organisations, and on the design of public policies. The EU Horizon 2020 programme, for instance, instantiates a system of innovation for the companies of EU member countries.

In this section I cover prior work around innovation and converge towards innovation produced by users and firms. I am particularly interested in user innovation and in SME innovation, in which companies engage in Open Source Software (OSS) practices in the creative industries. In the following section, I will cover some of the fundamental work that has been developed in the tradition of Innovation Studies, which I will use to contextualise and inform my own work.

2.2.1 Toolkits for User Innovation

One important and extensive body of work is User Innovation (von Hippel, 1986, 1994, 2001) that von Hippel detailed in his book "Democratizing Innovation" (von Hippel, 2005). User innovation focuses on user-centric innovation and assumes that users of products and services are empowered to innovate autonomously as a result of computer mediated capabilities and communication. As such, this stream of thought is more distant from firm-centric innovation processes, such as Open Innovation (Chesbrough, 2006) and traditional processes where the manufacturer role is mainly to identify users' needs and provide for them, fulfilling them with their products.

von Hippel (2005) defines User-Innovation Toolkits as "integrated sets of product-design, prototyping, and design-testing tools intended for use by end users" that enable them to "design high-quality, producible custom products that exactly meet their needs" (p. 163). User-innovation toolkits support an approach for product innovation and development, in which manufacturers "outsource key need-related innovation tasks to their users" (von Hippel, 2001, p. 3) rather than engaging in expensive efforts to understand user needs in detail for their internal product development process. According to von Hippel, this is achieved by providing the users with tools that grant them problem-solving capabilities and freedom to innovate, allowing them to design and develop their products in an iterative fashion, and to learn by doing and through trial-and-error, within the constrains of the solution space provided by the manufacturer.

The rationale behind user-innovation toolkits (von Hippel, 2001) is one the main kernels of von Hippel's theory of User Innovation. The two other premises of his theory are explained as follows:

a) *Lead users* – von Hippel (1986) identified lead users as a typology of users that are ahead of trends in their user groups and able to anticipate general demand and identify specific market needs. Lead users also have the technical abilities and great incentives to engage in modifying products or developing innovative solutions that can fulfil these needs. Further, von Hippel found that the intensity of these user characteristics correlates highly with the economic value and commercial attractiveness of the user innovations, which many manufacturers strive to find and commercialize.

b) *Sticky information* – von Hippel (1994) explains the shift of innovation from the corporate firm towards users with the theory of "information stickiness". He defines "information stickiness" in the context of product development as the incremental expenditure required to transfer a unit of information from one place to another, in such a form that is usable by a given information seeker. "Sticky" information emerges as a concept to characterize information that is difficult to acquire, transfer, and ultimately apply. It can be found on both sides of manufacturer and user. On the side of the manufacturer this concerns technical and solution specific knowledge. On the user side, it concerns need-related and product usage information; this information is typically obtained through ethnographic studies and other user-centric methods (such as focus groups, pilot studies, etc.) and traditionally, it is used to feed the manufacturers' development process.

However, von Hippel found that need and context-of-use information, which is typically located at the users' site, is very costly for the manufacturer to acquire, transfer, and ultimately use. The difficulty arises in providing an accurate and complete specification that covers all the details and subtleties. This usually entails an expensive and iterative cycle of problem-solving shifts between user and manufacturer that endures until a joint solution satisfactory to both parties is reached. He argues that this places the locus of problem solving on the side of the users, giving them the potential to innovate in the most valuable directions.

According to von Hippel (2001), *User-Innovation Toolkits* (UIT) first emerged in the context of custom integrated circuit (IC) design industry. This industry had a great efficiency problem by then, which was failing to understand the user needs accurately and in detail at the start of a product design project. This caused extremely high costs and motivated the introduction of user-innovation toolkits. The innovation step that lead to the creation of these toolkits was informed by the insight of Mead and Conway (1980)—cited in von Hippel (2001)—that custom design of ICs could be partitioned into solution-related and need-related subtasks.

On one hand, UITs isolated and standardized the solution-related information about fundamental elements of IC chips. On the other hand, UIT assigned the 'need' information and associated tasks to users, i.e. the way of interconnecting the standard IC elements, for building specific applications and product. LSI Logic introduced what is considered the first UIT in the 1980s (von Hippel, 1994). The LSI toolkit consisted on a set of proprietary software tools for custom IC design that allowed customers to design their own circuits by themselves. According to von Hippel, the results of this process were so impactful that these turned LSI into the top IC market player at the time, forcing its competitors to follow its lead and introduce their own IC design software tools for their customers. While the LSI software toolkit was very specific to the IC industry, von Hippel (2001) proposes that user-innovation toolkits "will eventually spread to most or all producers creating custom products or services in markets having heterogeneous customer needs" (p. 4). They can be developed as specific to any given product or service type, and production system.

UITs increase the user's creative potential by enabling them to modify and improve products while simultaneously constraining their improvements to actions and language that are compatible with manufacturers' processes. This does not entail, however, that the production system is locally or strictly under the manufacturer's control, as Von Hippel exemplifies toolkits that where development is at the locus of the user (e.g. software, websites, game development toolkits). Von Hippel also presents the attributes and benefits of UITs, and how they work:

- They allow users to engage in need-related problem-solving tasks, in learning through doing and trial-and-error rounds—toolkits should support users in problem-solving practices for product design and development, and the need information should be concentrated in a small number of simple tasks. Toolkits should provide creative freedom to users by allowing them to assess initial design choices and improve iteratively upon them.
- They offer a solution space that contains designs that the user wants to create—the creative freedom that a toolkit offers is in direct relation to the dimension of its solution space and production system. A toolkit with a small solution space will be able to combine a small set of pre-designed and special purpose options. A larger solution space could be made available with building blocks and capabilities to combine and manipulate them.
- Should be user friendly in order to provide good autonomy while requiring little specialised training—user-friendly toolkits should empower users to use their own skills and familiar design languages, minimizing the need for specific training. This will also reinforce the creative freedom by allowing users to focus on the creative aspect of problem solving and of designing the function that they want to incorporate in the product or service.
- Should provide libraries of standard modules to be readily integrated on custom designs. Pre-design modules in a toolkit provide great value to the user, allowing them to focus on the novel aspects of the design for their specific problem.
- Should ensure that the custom design is producible by the manufacturer without any additional modifications. A toolkit for user innovation should enable a successful translation from the design language into the production system language.

von Hippel (2001) also delimited the conditions in which innovation toolkits are able to provide the best value; user-innovation toolkits are most well suited for types of products and services in which there are high requirements of customization by users; also the highest value can be delivered when users "have need information that is sticky" (p. 22), that grows outdated quickly, and when "they must engage in learning by doing to clarify what they want" (p. 22) and meet their needs in a more accurate and complete way.

von Hippel has abstracted user innovation toolkits which can specialize for a wide range of domains, production systems and products. The IC industry provided the seminal example, but many other fields like custom food design, software and gaming industry have gained immensely with the use of toolkits (von Hippel, 2001). Moreover, von Hippel predicted that the adoption of toolkits will increase over time as tools for user-centric innovation, given the increasing trends around of personalisation and customization of products.

A more recent example of a user toolkit for innovation is the Apache Web server (Franke and Von Hippel, 2003). It has a modular architecture which is often tailored by its users, the Apache webmasters, to their heterogeneous needs. When users share these customisations both other users and Apache's developers benefit. This enables a more effective refinement of Apache's platform core, based on users' configurations.

Another well-studied application of a toolkit for user innovation and example of the OSS strategy of donated complements in the music technology domain is of that of Sibelius (Flowers and Voss, 2013), a standalone music notation software. Flowers and Voss analysed the plugin architecture of Sibelius, which enables users to develop their own plugins and extend its base functionalities. As in the previous approach, motivation of external collaboration is a key issue here.

Jeppesen and Frederiksen (2006) study the context in which firms use online user communities to reinforce their innovation process. Firms can benefit from online user communities by monitoring the feedback on their product, from getting ideas and picking on freely revealed user innovations to integrate and extend their product. Jeppesen and Frederiksen investigate how the users' personal attributes such as work-related status, "leading edgeness" in the field, and reputation mechanisms impact the users' participation in the community and innovative contributions.They conducted a case study where they focused on the online user community hosted by Propellerhead Software, a software firm that develops Rebirth and Reason, two popular computer-controlled music instruments.

According to Jeppesen and Frederiksen (2006), Propellerhead's first product, Rebirth, was cracked eight months after its release; hackers started to customize Rebirth by integrating their own sound samples and graphics. Propellerhead maintained a positive attitude towards user modifications and decided to open up parts of the code to support 'mods'; later on they gradually structured the software to enable user innovation by adopting user innovation toolkits (von Hippel, 2001). These events led to the emergence of a user-organized community, and subsequently, to the official community, which is hosted on Propellerhead's site. Jeppesen and Frederiksen report on the details of the whole process of collaboration and innovation between Propellerhead and its user community, and present their conclusions:

- Innovative users are likely to be hobbyists in the field in which they innovate, because professionals (professional musicians that use the software) are less motivated to engage in innovative unpaid unwork. Innovative users are often IT professionals who bring their expertise to use in their hobby field, and that share their innovation because they are not in competition and have nothing to lose. Furthermore, Jeppesen and Frederiksen report that 33% of the innovative users prefer to build on and extend existing innovations instead of having to start from scratch.

- Firm recognition has the biggest impact on why innovative users are motivated to participate in the community and freely reveal their innovations; this factor seems to be stronger than peer recognition since having the firm recognise the competence of the innovative user is the best way to propagate it to their peers. Jeppesen and Frederiksen suggest that this provides an opportunity for management to design their approach on how to employ firm recognition to motivate users (e.g. promote the best user innovations).

- Lead users are most likely to contribute with innovations in user communities. Here Jeppesen and Frederiksen corroborate von Hippels' theory on Lead users, and the importance and high-quality of their innovations. They also find that innovative users in the context of user communities most probably generate incremental-type innovations. Jeppesen and Frederiksen called for further research that examines how firms structure technologies for innovation (toolkits) and govern their community.

2.2.2 Human-Centred/User-Centred Design

In "The Inmates Are Running the Asylum", Cooper (1999) takes an interaction design perspective to discuss how software projects often lead to a product that is difficult to use and therefore compromised. He argues that this happens mostly in projects that have a high bias in terms of the type of effort that is invested (engineering vs design). Cooper argues that an imbalance happens when the focus on technical constraints and possibilities prevails over the focus on user goals. Alternatively, Cooper argues that this imbalance can also happen when visual designers are allowed to "run the show", in which the project might lack the sufficient robustness or underachieve the functionality that was planned for. To tackle this, Cooper proposes methodology for solving this problem. He proposes initiating the process with designing how a product will be used, then program the system, perform user and bug testing, and then optimise iteratively.

User-centred design (UCD), a term coined by Norman and Draper (1986), is a design approach based on understanding users, their tasks and environments. UCD has been characterised as "philosophy and methods, which focus on designing for and involving users in the design of computerised systems" (Abras et al., 2004, p. 12). When adopting UCD, we look for ways to better understand users, their characteristics, skills and behaviour in specific tasks. Users are at the centre of this process. They are involved at an early stage and throughout the design process. This enables the design team to communicate and negotiate a better and shared understanding of the right problem, and to reason about and inform design decisions for the right solution (Monk, 2007). Norman does note that in UCD, however, "even though ideal can seldom be met in practice, it is always good to aim for the ideal, but to be realistic about the time and budgetary challenges" (Norman, 2013, p. 239).

According to Ritter et al. (2014), UCD has a broader focus, greater emphasis on the user, and lesser use of formal methods for requirements gathering and specification than other approaches, such as human factors and ergonomics, or socio-technical systems design. They claim that adopting a user-centered approach can help to address important design problems and lead to systems that are more useful, usable and satisfying. A UCD approach can help designers overcome errors or issues that are based on using their personal assumptions, experience or intuition.

Furthermore, Ritter et al. (2014) argue that, on one hand, through the iterative and flexible cycle of user-centric development and evaluation, design gets refined and problems can be mitigated from the final product, leading to financial savings. On the other hand, the costs of the UCD process are usually pinned to two main aspects: the success of the final outcome, where "usability is neither a necessary nor sufficient condition for success" but "the lack of usability can be a sufficient reason for failure" (p. 14); and the return on investment (ROI)—the extent to which the time and effort spent doing user research provides worthwhile benefits—which is usually difficult to measure and to manage but is highly valued (Ritter et al., 2014). Holtzblatt et al. (2004) also refer to the ROI as an important criterion for adoption of UCD practices and as a reason for organisational resistance.

In order to avoid complex, time consuming and expensive techniques, Nielsen (1994) proposes applying discount usability methods, such as user and task observation, scenarios, simplified thinking aloud and heuristic evaluation, to design problems with a well-defined user population and set of tasks. Monk (2007) proposes lightweight techniques that can be easily picked up and applied effectively (i.e. learned in one day, only taking person-days to apply).

2.2.3 HCI perspectives and approaches to Innovation

Recently, some renowned HCI theorists (Greenberg and Buxton, 2008; Norman, 2010a,b) have expressed concerns about the capabilities and limitations of HCI for innovation purposes, and advocate for change to overcome these limitations. Greenberg and Buxton (2008) argue that usability engineering, a subfield of HCI, may very well support incremental improvement of existing products, but fails at supporting the design of new products and breakthrough innovations. They argue that iterative requirement gathering, prototyping and evaluation might be harmful, by sabotaging early creative ideas and visions, inducing confusion between scientific and commercial value, or hindering the future adoption of products overall.

Norman (2010a) also reinforces this point, stating that design research is good for improving existing products but useless for breakthroughs. He argues that recent innovations have been driven by a technology push rather than by user needs. Later on, Norman (2010b) explains that the shortcomings of HCI research for innovation purposes are most likely due to their different nature, goals and culture: where research is about abstracting scientific understanding of the use of technology, innovation practice takes a pragmatic and business-oriented approach towards product development. Inspired by translational science, Norman introduces the term 'translational development' as follows:

> *A discipline in which intermediaries translate research findings into the language of*

> *practical development and business while also translating the needs of business into issues that researchers can address [...] Translational developers are needed who can mine the insights of researchers and hone them into practical, reliable, and useful results. Similarly, translational developers must help convert the problems and concerns of practice into the clear, need-based statements that can drive researchers to develop new insights.*
>
> (Norman, 2010b, p. 12)

These contributions have inspired a recent track on CHI about innovation (Chilana et al., 2015a). The discussion focused on new developments that may extend beyond typical HCI research methods such as the introduction of business and product development-oriented approaches. Some of these proposals, which are reviewed below, suggested extending the focus beyond end users to encompass a variable set of stakeholders. Other proposals suggested activities around product development and strategic decision-making for marketing, beyond the more traditional activities of HCI such as ideation, participatory development, or evaluation and validtation of the system.

Holmquist (2013) takes an HCI perspective on innovation, introducing grounded innovation as methodology which is grounded both in technology and in user studies. This approach intends to maximise what Holmquist considers to be the axis of innovation—invention and enquiry. For Holmquist, invention is driven by the pursuit of additional benefit throught novelty and originality, in contrast to enquiry, which provides a grounding based on the possibilities and opportunities of the real world and data gathered from users. He suggests that these two components are ineffective if in isolation, but when combined together they may provide a better framework for successful innovation. He also discusses how different methods fall differently along these two axes and how each of them emphasises different levels of invention and enquiry. This is illustrated by comparing conceptual art projects, which display a great level of invention and idea generation, to pure studies, which focus on enquiry in detriment of invention.

Holmquist (2013) argues that HCI methods such as user-centred design (UCD) offer a balanced combination of invention and user enquiry, by enabling the adoption of user needs as the foundation for building new technologies. He also argues that HCI provides methods for invention (such as brainstorming, body storming and bootlegging) and for user-oriented enquiry (such as focus groups, interviews, questionnaires and empirical studies), which can help to ensure the fitness and usability of the inventions. He also differentiates between innovation and design, where design is a problem-solving effort with a clear goal, usually bounded by a specification (such as a client brief) and the constraints imposed by the available materials. Contrastingly, for Holmquist innovation has no predefined specification or goal, and strives to transcend the limitations of materials. However, his perspective grows very attached to the research stages rather than product development.

Lindtner et al. (2014) present a longitudinal ethnographic study in which they document the current design and innovation practices that are happening around the DIY maker movement, and in sites such as hacker spaces, hardware start-ups and incubators. Lindtner et al. review the origins of the DIY maker movement, its political discourse around economic recovery and innovation, and the literature on HCI applied to maker practices. They also analyse how the confluence of new funding models such as crowdfunding, the emergence of physical spaces such as hackspaces, and new platforms, tools and publications are propelling this movement into the mainstream and becoming professional practice.

Lindtner et al. (2014) reflect on the role HCI takes in these processes of innovation and on their implications for this field. Among the findings of the research, they conclude that there is a new general interest in returning to physical materials and experimentation with new processes of production and entrepreneurship, such as open collaboration and rapid prototyping. They also report on how these processes appear to sidestep other typical HCI and R&D processes from more traditional and established research organizations, such as academic and industrial R&D labs, which many consider slow operators and often stuck in IP quarrels. Lindtner et al. suggest that this alternative path of development provides grounds for new HCI research around new models of interaction design and technology production that bridge theory, design, manufacturing and entrepreneurship.

Chilana et al. (2015b) present a case study of a research project around software engineering and HCI. The technology that emerged from this research transitioned to a successful innovation that was exploited by a start-up funded by venture capital. They document the evolution of the research prototypes, of product development activities and the establishment of the start-up business. The paper presents the motivations and rationale for adopting different methods during the process of innovation. They highlight the trade-offs made between typical HCI methods such as user-centred design and what they have coined as “adoption-centred design”.

Chilana et al. used HCI methods for designing and building the prototype and validating it with users, leading to research outcomes. For commercial product development, they used adoption-centred design focusing on market innovation methods, such as business models, marketing, productisation, stakeholders, value proposition, market entry barriers and B2B adoption. Based on their successful experience, Chilana et al. call for further informing efforts that will help transform HCI technology research “from a source of ideas to a source of commercially disseminated solutions that create widespread value”. They argue about the need to investigate adoption-centred design further and about its integration within HCI research, suggesting possible ways on how to achieve it and the benefits of such an approach:

> *A more explicit adoption-centred approach to research might increase the chances that an investor, entrepreneur, or prospective employee would see business opportunities in HCI research. Combined with other systemic changes, such as more extensive and rapid publicity of research innovations for the public and greater awareness of university intellectual property policy, an adoption-centred focus in HCI research might lead to a discipline of HCI technology transfer.*

(Chilana et al., 2015b, p. 1757)

2.2.3.1 Human-centred Infrastructural Software Development and Toolkits

Application programming interfaces (APIs) are the developer-facing constituents of infrastructural software—that is, software that supports the development and operation of other software (Edwards et al., 2003). The use of APIs is a standard practice in software engineering given the importance of modularity and reusability (Fowler, 2004). While their use can provide potential savings in time and effort for common tasks, comparing this payoff to the cost of programming a custom solution is not straightforward; developers making an informed decision about adopting an API may have to consider their previous experience with that API and API domain and with its conceptual model, design cues, and design patterns (Blackwell, 2002). Learning about these elements for the first time may be challenging. The structure of API documentation may significantly impact on this learning, e.g. by the prior conceptual knowledge it assumes, the application scenarios it targets, the code examples and other learning resources it provides (Robillard and Deline, 2011).

Designing an API is a challenging task. The API must meet users' technical requirements (e.g. around performance, robustness, correctness, stability, security) (Henning, 2009). The API must be useful, providing an appropriate set of features for a space of potential client applications (Edwards et al., 2003), usable (Myers and Stylos, 2016) and provide effective learning resources (Robillard and Deline, 2011). There exist different approaches to API design. A designer-centric approach to API design is mostly based on the designer's taste or aesthetics (Venners and Eckel, 2003); this can be successful when the designer has extensive experience in both API design and in the API application domain. Approaches based on API design heuristics use empirically-based knowledge that has been compiled into prescriptive guidelines or recommendations—e.g. technical books about API design (Cwalina and Abrams, 2009; Tulach, 2008). There is, however, contradicting empirical evidence about the usability of certain API design heuristics (Myers and Stylos, 2016). User-centred design (UCD) approaches inform and drive API design with usability data—e.g. (Clarke, 2010; Myers and Stylos, 2016). This can be useful to counteract misleading assumptions about API users that are not represented within the API designers' group. There are claims, however, that usability approaches can be excessively focused on specific design features and may fail to deliver in a more holistic way (Venners and Eckel, 2003).

Myers and Stylos (2016) provide a comprehensive review of different methods to measure and improve the design of APIs. One traditional approach is API peer reviews (Wiegers, 2002), where technical peers examine and give feedback. An alternative is the API Concepts framework (Scheller and Kühn, 2015), which automatically evaluates both the API and samples of client code, considering user characteristics (i.e. learning style, programming experience, etc.) among the critical factors of evaluation. Other methods have been adapted from traditional HCI and usability engineering, including empirical and task-specific evaluation techniques (e.g. think aloud protocol, cognitive walkthrough) as well as heuristics-based techniques (Myers and Stylos, 2016).

Other approaches to API evaluation are based on the Cognitive Dimensions (CDs) of Notations framework (Green, 1989; Green and Petre, 1996). CDs are a “broad-brush” evaluation set of tools that support discussion about the design trade-offs of information structures. API studies based on the CDs typically either use the questionnaire originally developed by Blackwell and Green (2000), or a shorter or partially refactored version specialised to a specific domain. For instance, the original CDs questionnaire was used by Austin (2005), who assessed the usability of a functional shading language for graphics programming, and by Diprose et al. (2017), who assessed the abstraction level of an end-user robot-programming API.

Clarke and Becker (2003) derived a framework from the original CDs to characterise specifically how API design trade-offs meet the expectations of the API users. They applied it to evaluate Microsoft .NET class libraries. Watson (2014) applied Clarke’s framework for improving API documentation planning. Wijayarathna et al. (2017) adapted the questionnaire from Clarke and Becker (2003) to evaluate the usability of a cryptography API.

You must go to the scene of action, first, because men put more faith in their eyes than in their ears, and second, because the way is long if one follows precepts, but short and helpful, if one follows patterns.

— Seneca

Chapter 3

Methodology

The main purpose of this chapter is to introduce the reader to my research methodology design. Given my stated research questions, I have chosen to adopt a mixed methods approach incorporating the following methodologies: Canonical Action Research (Davison et al., 2004), Action Design Research (Sein et al., 2011), and User-Centred Design (Norman and Draper, 1986). These approaches were used to interrogate the research questions through discussion, analysis and specific experimental studies, details of which can be found in chapters 4 and 5. As part of this process, frameworks rooted in these methodologies were synthesised both through collaboration with research partners (*UCD Actions* framework, Section 4.2) and analysis—i.e. in the case of the unified framework of von Hippel (2005), Amershi (2012) and Dudley and Kristensson (2018)).

The following section provides an overview of Action Research in order to justify the selection of a research focus, justify its adoption and provide guidance for the research design, and unveil the limitations of the methodological choice. The following sections cover the origins of Action Research, fundamental literature, the structure of research design, and the mapping of theoretical grounding with the practical interests of a client project. Also, some recent methodological developments in the form of derived frameworks are reviewed—Canonical Action Research and Action Design Research.

3.1 Rationale for research strategy and methodology design

I chose Action Research as the main research methodology for this thesis for several reasons. First of all, my personal involvement in RAPID-MIX with the role of research assistant hired a) to engage in collaborative research and development with project stakeholders and end users, b) to learn through action and contribute to practice with interventions and insights to inform design and development, and c) to contribute with new knowledge about the design and evaluation of new ML tools for creative developers and research community.

Second, the emphasis of the generic AR framework on problem solving, collaborative action-based learning, and generation of empirically-based knowledge for academic and practitioner communities alike (Section 3.2). This made me consider it as an adequate methodology, capable of providing necessary support for my research, particularly as AR has been considered complementary to both UCD and participatory design practices (Foth and Axup, 2006) and used in HCI research (Section 3.3). AR promised an effective approach for obtaining a more systematic and holistic understanding about the client project and organisational setting, and the different interventions, in contrast to narrower or more localised methodologies—e.g. using UCD only or experimental usability studies.

Also influencing my methodological choice were the calls for new methodologies to support new and more relevant problems in HCI—for instance, for addressing innovation more holistically (Chilana et al., 2015a,b)—and my motivation to include such methodologies within the scope of this thesis. Nevertheless, the literature informed me of caveats regarding the generic AR framework (Section 3.2). I decided to take additional measures—i.e. unit of analysis, multiple iterations, grounding in theory—to mitigate those issues and ensure the necessary rigour, reliability and transferability of my research outcomes. With these methodological issues in mind and guided by the purpose of making CAR and ADR compatible, complementary and useful to practice, and simultaneously consistent in epistemological terms, I decided to adopt their principles to configure my research to the client project requirements.

In Section 3.4, I review the principles and additional criteria that CAR proposes (Davison et al., 2004) for mitigating the generic AR framework shortcomings in rigour, reliability and transferability. CAR proposes to achieve this by emphasising the importance of having a unit of analysis, grounding research in theory, and employing multiple iterations. I adhered to these principles by a) framing the process of design and evaluation of RAPID-MIX API as my unit of analysis; b) grounding my research on the theory of User Innovation (von Hippel, 2005); and c) planning and designing AR cycles with cross-sectional studies focused on informing or evaluating the design artefact.

Additionally, in my methodological approach I am employing ADR (Sein et al., 2011) as I found that it could also provide adequate support to my methodological design (Section 3.5). ADR promotes balanced support between innovation through design and knowledge emergence, and intervention in real settings. ADR can provide methodological scaffolding to study the emergence of a theory-ingrained IT artefact within an organisational setting. As such, the elements of ADR map adequately to the context and characteristics of both RAPID-MIX and RAPID-MIX API. ADR provides a sound structure for the analysis and evaluation of the design artefact, directing the inquiry to its nature, utility and degree of innovativeness. In Chapter 5.2, I apply the ADR principles to my research problem formulation. In Chapters 5 and 7, I apply ADR elements to structuring and scaffolding the different studies of design and evaluation of RAPID-MIX API.

3.2 Action Research origins and epistemology

Action Research (AR) is an umbrella term associated with a vast body of research methods that have emerged in the context of social sciences. These methods share a focus on organisational problem solving through intervention and positive change, while simultaneously contributing to scholarly knowledge.

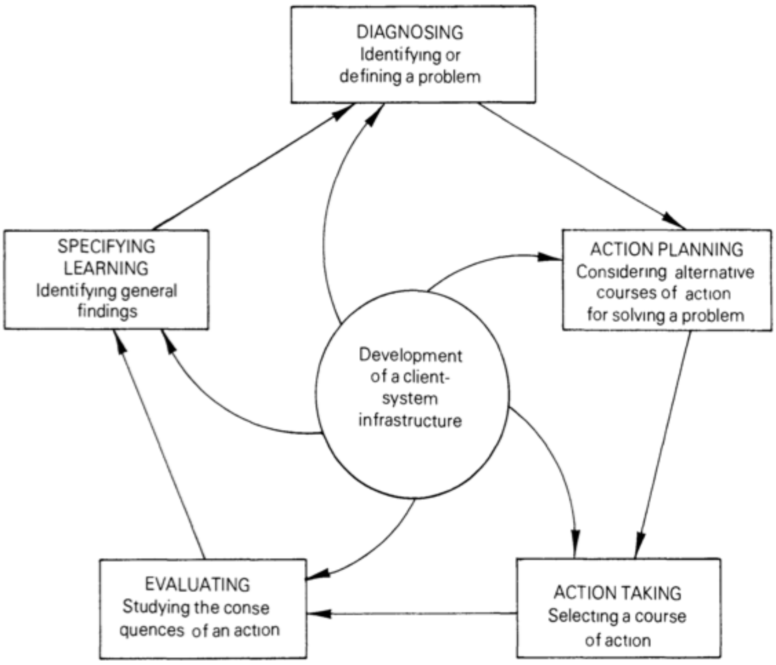

DIAGNOSING
Identifying or defining a problem
ACTION PLANNING
Considering alternative courses of action for solving a problem
ACTION TAKING
Selecting a course of action
EVALUATING
Studying the conse quences of an action
SPECIFYING LEARNING
Identifying general findings
Development of a client-system infrastructure

Figure 3.1: The Cyclical Process of Action Research (Susman and Evered, 1978, p. 588). Permission to reproduce this figure has been granted by *Administrative Science Quarterly*.

AR origins have been credited to two independent research efforts: a) the work of Lewin (1946) around group dynamics and social change in the US, and b) the operational research of Tavistock Institute of Human Relations in London, around psychosocial war disorders diagnosed in veterans from World War II (Trist, 1976).

One definition of Action Research often used in contemporary literature is that of Rapoport (1970):

> *Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework.* Rapoport (1970, p. 499).

This definition is grounded in the social sciences from which Action Research originated but it has been adopted into different fields. After becoming established in the social and medical sciences, the popularity of AR grew in Information Systems (IS) research in the late 1990s, a field with a strong socio-technical emphasis.

Action Research provides a framework based on a multi-step iterative cycle that involves the identification of a general problem to be solved, or an improvement opportunity at the client organisation. As Lewin (1946) first conceived, the process is “a spiral of steps, each of which is composed of a circle of planning, action, and fact-finding about the result of the action” (Lewin, 1946, p.38).

Each cycle is composed of steps that focus on the identification of practical problems, a solution to those problems, action taken for change by means of implementing the solution, and reflection on the part of the research body which will eventually be transferred to the client as knowledge. This is then followed by another round of identification and solution of problems, new reflection, and so on. One often cited diagram of the AR cyclical process, which has been credited to Susman and Evered (1978), is shown below in Figure 3.1.

Kock (2007) proposed the use of AR from an IS research perspective, in the study of human behaviour toward technologies, arguing that it can be used in investigations aimed at testing hypotheses in a post-positivist fashion[1]. For Kock (2014), the AR framework leads to the identification of clearer patterns through repeated observations in the various iterations of a technology-related inquiry. This could entail the introduction of new technologies in an organisation and studying the impact of the technology in that organisation. The subsequent iterations of the AR cycle can take place in the same or in a different organisational context (e.g. a different department or company).

> *Conducting organizational AR involves helping an organization solve its problems and become ‘better’ in terms of some of its key attributes such as productivity, the quality of their products and/or services, and working conditions. At the same time, AR involves collecting, analysing, and drawing conceptual and theoretical conclusions from organizational research data.*
>
> (Kock, 2007, p. 98)

In the context of IS, Action Research has been described as “a post-positivist social scientific research method, ideally suited to the study of technology in its human context” (Baskerville and Wood-Harper, 1998, p 235). The method produced “highly relevant research results, because it is grounded in practical action, aimed at solving an immediate problem situation while carefully informing theory” (Baskerville, 1999, p. 2).

Susman and Evered (1978) argued the deficiencies of positivist science for generating useful knowledge to solve problems in organisations. They proposed Action Research as a methodology able to correct these deficiencies. They characterised the epistemological tenets of AR and compared them with the positivist stance, more commonly adopted in the scientific community (Table 3.1).

Action Research has been criticised regarding its methodological rigour, relevance, emergent ethical issues related to an unclear distinction from consulting, and to the potential asymmetry of contribution between the action and research components. Baskerville (1999) identified a series of limitations and issues that the researcher implementing an AR design may face, such as:

[1] According to Trochim (2006), “post-positivist critical realism recognises that all observation is fallible and has error and that all theory is revisable.”

Table 3.1: Comparison of Positivist science and Action Research Susman and Evered (1978, p. 600)

Points of comparison	Positivist science	Action Research
Value position	Methods are value neutral	Methods develop social systems and release human potential
Time perspective	Observation of the present	Observation of the present plus interpretation of the present from knowledge of the past, conceptualisation of more desirable futures
Relationship with units	Detached spectator, client system members are objects to study	Client system members are self-reflective subjects with whom to collaborate
Treatment of units studied	Cases are of interest only as representatives of populations	Cases can be sufficient sources of knowledge
Language for describing units	Denotative, observational	Connotative, metaphorical
Basis for assuming the existence of units	Exist independently of humans	Human artefacts for human purposes
Epistemological aims	Induction and deduction	Conjecturing, creating settings for learning and modelling of behaviour
Criteria for confirmation	Logical consistency, prediction and control	Evaluating whether actions produce intended consequences
Basis for generalisation	Broad, universal and free of context	Narrow situational and bound by context

a) becoming "embroiled in the problem setting, and los[ing] contact with their obligations to develop general knowledge about related theories" (p. 26) or by misleading clients who are expecting more of a consultative role, which may also raise associated ethical issues related to the financial support.

b) problems with the collaborative nature of AR and lack of control, which "diminishes the researcher's ability to control the process and the outcomes of the research" (p.26), and may prevent researchers from "'pick[ing] and choos[ing] the problem they wish to investigate".

c) finding "that the theoretical emergence twists the research in an entirely different direction" (p.26) from the problem suitable to their predefined research program.

In the same argument, Avison et al. (2001) discuss how aspects of control in an AR project (i.e. initiation procedures, authority structure and processes within the project, and the formalisation degree) can affect the rigour and, more drastically, the success of an AR project. Avison et al. argue that the inability to negotiate the control structures in AR projects has been considered compromising to the success of an AR project.

In order to address these problems, Kock (2007) took initial steps to reconcile AR methodology with more general research methods. He identified three main categories of problems (or 'threats') that undermine the potential for AR to generate relevant outcomes to both the industry and scholarly knowledge: 'uncontrollability', 'contingency', and 'subjectivity'. Kock proposed three antidotes to resolve these methodological 'threats': (a) unit of analysis, (b) grounded theory and (c) multiple iterations. Later on, Davison et al. (2004) proceeded further to tackle these issues. They developed Canonical Action Research as model to ensure and assess the rigour and relevance of AR in IS research. I will examine this methodological development and one other in more detail in Sections 3.4 and 3.5.

3.3 Action Research in Human-Computer Interaction

Several HCI researchers have employed AR in their work. For Hayes (2011), AR is "a class of methods and approaches for conducting democratic and collaborative research with community partners" (p. 1). Hayes considered that AR "offers HCI researchers theoretical lenses, methodological approaches, and pragmatic guidance for conducting socially relevant, collaborative, and engaged research." (p. 1). She also considered that the action-based, collaborative and iterative nature of AR, and most importantly, the level of scientific rigour that it provides, permits the transferability of research findings to the research community.

Foth and Axup (2006) also considered the notions of reflective learning and change through action that AR includes, complementary to both UCD and participatory design practices. Dix (2012) listed other research projects where AR had been employed and valued for HCI research.

Responding to an emergent interest from the HCI community, Kock (2014) directs the attention to the desirability of AR for HCI and to the low percentage of submitted works using this methodology. Kock argues that AR has been most valued by external funding agencies, such as the National Science Foundation in the US and the European Commission, and by large tech conglomerates, such as Google and Microsoft, which have been focusing on practical research approaches such as AR, to enhance the credibility of the research findings.

Over the years, AR has been developing through derivations, which seek to strengthen its rigour by improving on its methodological limitations, or specialisation to the characteristics of a specific domain. One example of the former case is the Canonical Action Research (Davison et al., 2004). Another methodological framework that derived from AR is Action Design Research (ADR) (Sein et al., 2011), an example of the latter.

More recently, Bilandzic and Venable (2011) elaborated further on ADE and proposed Participatory Action Design Research (PADR), which borrows from Soft Design Science Research and

ADR for studying urban informatics. Urban informatics studies the urban life experiences created from novel real-time technologies and uses techniques that originate from the fields of ubiquitous computing (Ubicomp) and HCI. PADR was developed to address the specific cross-disciplinary needs and research contexts of urban informatics. Bilandzic and Venable adapted their techniques and approaches with more ethnographic and participatory qualities and combined them with Design Science Research, which they claim more effective for analysis and problem solving, and more typical of design methodologies.

3.4 Canonical Action Research

Canonical Action Research (CAR) (Davison et al., 2004) has been adopted extensively in IS. CAR is a prescriptive framework that derives from AR and which encompasses a set of interdependent principles to guide practice with evaluation criteria for process review. Davison et al. suggested that a researcher who adopts AR should adhere to these principles (See Appendix A). First, these principles ensure the rigour and relevance of research, by providing a set of criteria to counteract previously identified issues of the generic AR framework. Second, they facilitate "the clear and systematic presentation of ideas and findings, at the same time helping researchers to justify their choices of action, their contributions to knowledge and their conclusions" (Davison et al., 2004, p.2). This set of principles is described as follows:

- *Research/Client agreement* (RCA) – the RCA agreement, which Davison et al. consider one of the foundations of the methodology, is established between researcher and research client and negotiates the entry of the researcher into the client system. In a way, they suggest, this process can be seen as a knowledge transfer process as it informs the client about the role and responsibilities of the researcher, the process mechanics, benefits and disadvantages, but also the goals, measures, and data collection and analysis methodologies. This aims to foster trust among stakeholders of the client project, ensure their cooperation and reinforce the validity of research. Elements that should also be specified and agreed on at this stage include the a) expected duration, b) the research focus and how it can be adjusted throughout the project, and c) the commitment to implement changes. The RCA should be revised and updated should there be any major changes that occur during the process.

- *Cyclical Process Model* (CPM) – the CPM subsumes the general AR cyclical model, which has been described previously in section 3.2, although Davison et al. have introduced developments to the cycle. Figure 3.2 presents the CMP, which formalises the entrance into the client system (through the RCA), the exit of the project, introduces seven questions that guide the enforcement of the research quality criteria; they suggested the use of a spiral instead of a cycle to reflect how sequential interventions can support a gradual approximation to the core of the organisational problem. This sequential and iterative progression through the CPM ensures systematic rigour, although Davison et al. admit the need for flexibility

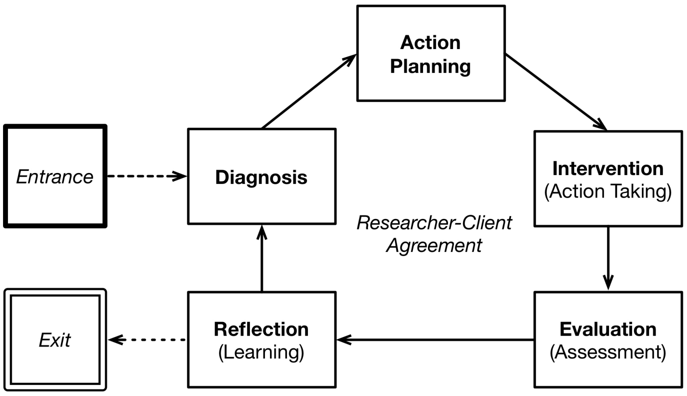


Figure 3.2: Canonical Action Research process model, adapted from Davison et al. (2004)

(through non-linear execution of the stages) in order to tackle issues and obstacles that may emerge at a given stage of a cycle. All stages should complete, and any case of deviation should be documented and justified.

- *Theory* – in order to ensure the relevance of research, Davison et al. recommend the articulation of a sound theoretical model to guide and focus research activities (e.g. data collection, analysis), evaluation of outcomes, and positioning of outcomes within the guidelines and traditions of a specific scholarly domain. This might be through a grounded theory that emerges from the diagnosis stage, or some adopted set of theories. Alternatively, Davison et al. admit the possibility of beginning the project with "theory-free action learning" (Davison et al., 2004, p. 74). However, preventing this last scenario may help to avoid a common pitfall of AR, the "irrelevant subject failure" (Avison et al., 2001, p. 30). In this situation, the client typically persuades the researcher to conduct a study which is relevant to its own interests but insignificant or irrelevant for the research community.
- *Change through action* – actions should address the problem and its hypothetical causes, as identified in the initial diagnostic. The researcher should plan the actions appropriately and design intervention for a positive change, and have the client approve them. The actions should be implemented in a specific context or environment, and they should also be documented, explained and justified as part of the assessment that measures the change that was obtained.
- *Learning through reflection* – the researcher undertakes the critical activity of specifying the implications for scholarly knowledge and practice. The researcher should use progress reports, keep the client informed and engaged in reflection about the outcomes of the actions. Reflection should focus on the extent to which the knowledge acquired from the actions helps to generate new theory, or to add to or refute a given existent theory. The knowledge obtained may uncover the need to address the same focus with more actions, or to apply the same actions to similar research domains. Reflection should also include the suitability of CAR as applied methodology to the domain.

3.5 Action Design Research

Action Design Research (ADR) (Sein et al., 2011) is another methodological framework that derives from AR and had its inception in the context of IS research. ADR was designed as an extension to complement another major methodological approach in IS—Design Research. According to Sein et al., Design Research (DR) develops prescriptive design knowledge through building and evaluating IT artefacts intended to solve identified classes of problems. Design Research promotes technical novelty and abstraction for developing knowledge, and evaluates and measures the relevance of technology artefacts by utility.

Sein et al. (2011) seek to reconcile DR and AR with ADR, by filling the gaps and complementing each approach, bringing the focus onto practical problems with theoretical relevance and supporting authentic intervention in an organisational setting. They justify adapting rather than adopting IS research methods by combining AR and DR, stating that new information systems should not be developed in isolation from the environment in which they would be used. They propose a tight coupling between the research activities of building, intervention, and evaluation, with participation of key stakeholders (e.g. researchers, problem owners, and system users). They recommend for this tight coupling to be implemented as a process that develops knowledge through intervention in an organisation, and building and evaluating an innovative IT artefact.

Sein et al. (2011) cite Orlikowski and Iacono (2001) extensively to characterise the IT artefact and the unit of analysis of ADR. According to Orlikowski and Iacono (2001), the IT artefact is an 'ensemble' which includes socially recognised amalgamations of hardware or software, inscribed with 'structures' of the organisational domain, as something that emerges through interaction between technology and an organisational context. The IT artefact is shaped by the interests, values, and assumptions networks and communities of stakeholders of that context, including developers, sponsors and users.

To study ensemble artefacts, Sein et al. (2011) propose ADR as a research method that accounts for both technological and organisational contexts, in which their interaction shapes the artefact via design and use, with the influence of both designers and users. ADR provides explicit guidance for combining building, intervention, and evaluation in a concerted research effort, in order to overcome what Sein et al. (2011) highlight as a major shortcoming of DR, which is the separation of building, intervention, and evaluation (BIE) activities and the sequential structuring that is introduced by stage-gate models[2]:

To address the problems of the nature of the artefact, its degree of innovativeness in relation to a class of systems, the sequencing and control that a sound evaluation requires, Sein et al. (2011) designed ADR to comprise a series of stages with guiding principles. Figure 3.3 depicts the stages and principles of ADR, which I briefly describe as follows:

[2] The *stage–gate model* is as sequential process with a series of stages where *gates* are used for senior management decision-making to decide on whether the product should continue to be developed. Cooper (1990) developed this model in the context of new product development and innovation management.

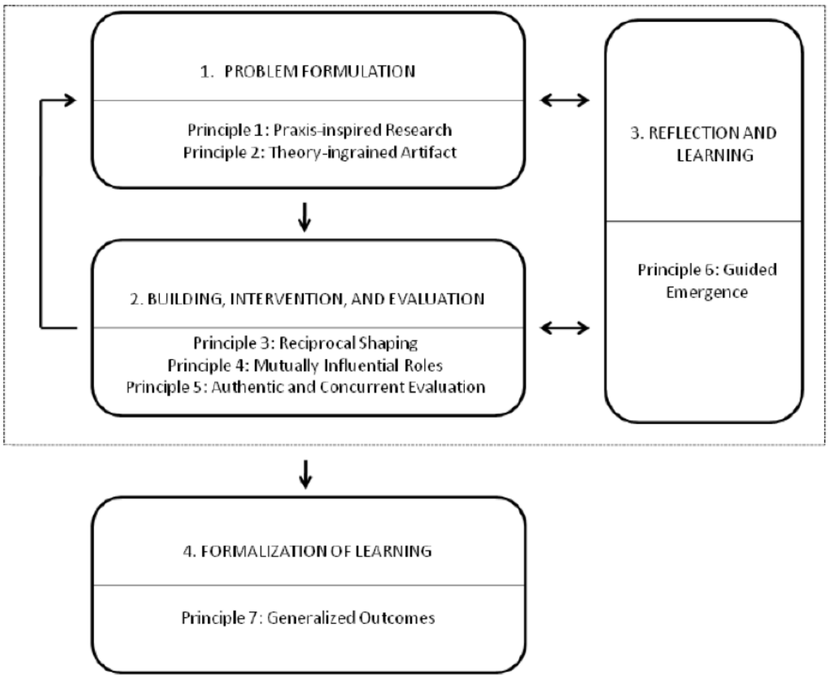

1. PROBLEM FORMULATION
Principle 1: Praxis-inspired Research
Principle 2: Theory-ingrained Artifact
2. BUILDING, INTERVENTION, AND EVALUATION
Principle 3: Reciprocal Shaping
Principle 4: Mutually Influential Roles
Principle 5: Authentic and Concurrent Evaluation
3. REFLECTION AND LEARNING
Principle 6: Guided Emergence
4. FORMALIZATION OF LEARNING
Principle 7: Generalized Outcomes

Figure 3.3: Stages and principles of the ADR methodology (Sein et al., 2011). Copyright © 2011, Regents of the University of Minnesota. Used with permission.

1. *Problem formulation* – The research opportunity and operational problem perceived within the organisation should be identified, articulated and scoped in order to start and drive the research efforts. Two principles are at stake at this stage: *Practice-Inspired Research*—"field problems [...] as knowledge-creation opportunities" (Sein et al., 2011, p. 40); and *Theory-Ingrained Artefact*—artefacts as carriers of theory, with theory structuring the problem, identifying solutions, and guiding design iterations.

2. *Building, Intervention, and Evaluation* (BIE) – this stage "interweaves the building of the IT artifact, the intervention the organisation and the evaluation" (Sein et al., 2011, p. 42) throughout the execution of the cycle. At this point, the locus of innovation or knowledge generation target must be defined from within a continuum that ranges from *IT-dominant BIE* to *Organisation-dominant BIE.* Figure 3.4 below illustrates the differences for these two types of BIE in an ADR process. While the former places emphasis on the creation of the IT artefact, the latter places emphasis on the organisational intervention. Where early prototypes and alpha versions provide for more incipient interventions to guide the design of the artefact, as it becomes shaped by the participating members' influence, it is finally evaluated in use settings. The BIE stage is supported by three principles: *Reciprocal Shaping*—"influences are mutually exerted by [...] the IT artefact and the organisational context" (Sein et al., 2011, p. 43), *Mutually Influential Roles*—"mutual learning among the different project participants" (Sein et al., 2011, p. 43), and *Authentic and Concurrent Evaluation*—"evaluation is not a separate stage of the research process that follows building [...] it is on-going" (Sein et al., 2011, p. 43).

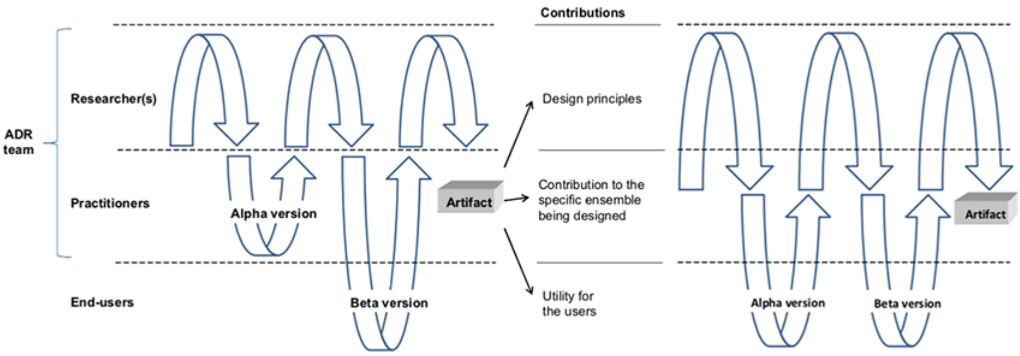


Figure 3.4: ADR schemas of an a) IT-dominant BIE vs b) Organisation-dominant BIE, adapted from Sein et al. (2011). Copyright © 2011, Regents of the University of Minnesota. Used with permission.

3. *Reflection and Learning* – analysis and reflection of the artefact and of the intervention results should follow in this stage, where the learning should be articulated in terms of chosen theories to inform refinement and redesign iterations. This stage is guided by the principle of *Guided Emergence* (where reflection upon the ensemble artefact should reflect the interplay between the preliminary design, its shaping by the organisational context, and the outcomes of evaluation. This is also where the inductive step begins to form, where there is a move "from building a solution for a particular instance to applying that learning to a broader class of problems." (Sein et al., 2011, p. 44).

4. *Formalisation of Learning* – as postulated by Sein et al., in this stage the learning that emerges should be developed "into general solution concepts for a class of field problems", shared with practitioners in the form of design principles that are articulated with the chosen theories, and disseminated to the scientific community. Thus, the principle on which this stage draws is that of *Generalised Outcomes*, where both the problem that was addressed and the solution that was found should be generalised. Here, design principles should be derived from the research outcomes.

3.6 Philosophical stance and positionality in the Action Research spectrum

The philosophical stance that is implicit in my research is post-positivism. According to Robson (2011), post-positivism concedes that the researcher can influence what is observed—with his/her goals, values, background knowledge, theories, etc. This opposes the positivist approach, in which the researcher should remain neutral and detached from the object of study. A post-positivist stance does retain the notion of objectivity, but recognises, nevertheless, that its pursuit should acknowledge and disclose the background knowledge and values of the researcher (Robson, 2011).

As previously explained in Section 1.2, the context of my research required my involvement and immersion in different work packages and activities of RAPID-MIX. A post-positivist stance is therefore compatible with the practical requirements of collaborative activities within RAPID-MIX and convenient to my research design.

In order to inform my research design decisions with adequate methods that strengthen the validity and consistency of research, I identified the philosophical stance underlying my research study. I am contextualising the methodology according to epistemological and ontological assumptions. The philosophical stance also determines the choice of methodology, and post-positivism considers both quantitative and qualitative methods to be valid approaches. In Section 3.1, I detailed the choice of an Action Research approach for the main methodology. In Section 3.2, I reviewed the tenets of Action Research and how this methodology is anchored in post-positivism.

Herr (2005) recommends for doctoral studies using AR the definition of the positionality of the researcher in the AR continuum. The positionality of the researcher in the Action Research spectrum is important to the epistemological, methodological and ethical framing and to expose how this influences decisions that occur throughout doctoral research (Herr, 2005). According to Herr, doctoral dissertations in AR "are often done by organisational insiders who see it as a way to deepen their own reflection on practice toward problem solving and professional development. In such cases, the researcher and the practitioner may be one and the same." (Herr, 2005, p. 47)

Such is the case with my involvement with RAPID-MIX as a participating action researcher. The invitation to participate in the project could be justified by a number of convergent factors associated with my background and professional skills. These factors can potentially contribute to the quality and effectiveness of my research and practice, and should be acknowledged:

1. My academic background and research experience in computer science, systems engineering and HCI—I was exposed to all the major areas of Computer Science, systems analysis and design. This included hands-on experience in different flavours of AI, including machine learning, evolutionary computing, and expert systems. After graduation, I interned in ALGORITMI, a research centre with a long tradition in Information Systems, and joined the UbiComp group. There, I worked as a research assistant and software engineer, and developed APIs and middleware for context-aware applications, and classifiers for object detection with computer vision. These technologies included the use of machine learning.

2. My academic background and research experience in management and innovation in the creative industries—I took a strategic management perspective to investigate how networked media technologies are supporting independent music artists to develop new business models and take disintermediated approaches within the value chain of the music industry. I also worked in CITAR, a research centre which focused on critical reflection of the processes and technologies for artistic creation. This experience afforded opportunities to learn about epistemology, research methodology design and critical thinking. Furthermore, I also had the opportunity to join a local consortium for technology transfer and innovation, and experience the challenges and shortcomings of the organisation of such processes.

3. My professional experience as a professional software engineer in R&D, working in different corporate environments, from research spin-offs to one start-up, through to a medium software enterprise—which contributed to my experience of different processes of technology transfer. I have been immersed in the closed model of innovation for a significant amount of time in my professional career. I have experienced the difficulties and inherent resistance that exist in engineering environments and enterprise software relating to the implementation of processes of user-centric research, and design for radical innovation.

4. My artistic background and personal interests in audiovisual creativity and musical expression—receiving a classical education in music and instruments, and joining bands of different musical genres which achieved different degrees of commercial success—also contributed to my notions of how different kinds of innovation (i.e., technical, social, cultural, etc.) are important and operate in general, and in particular, in the creative industries sector.

5. Finally, and perhaps most importantly, my status as PhD student and role of research assistant in RAPID-MIX. As a PhD student, I had the need to write a doctoral thesis and find a job that could both accommodate this goal and provide me with financial support. As a research assistant in RAPID-MIX, I was tasked with responsibilities in different work packages, as detailed in Section 1.2. These conditions determined my decision to conduct my doctoral research in convergence with RAPID-MIX project's goals, and to employ the principles of AR in order to ensure the viability and validity of my research, and to maximise its relevancy.

Although these factors contribute positively to my research, they may condition my perspective with potential biases. For instance, my previous experience with 1) using machine learning and designing APIs could lead to the attribution error and authority bias; my previous experience with 2) other research and innovation projects and 3) different corporate environments could potentially carry the biases of projection and pro-inovation, etc. Furthermore, the choice of AR as methodology determines a position of low-control—as observed and discussed by Avison et al. (2001) (Sections 4.5 and 5.9 3.2)—where the hierarchy and authority structure of the organisation setting can affect the rigour and validity of research of the AR project.

I engage these issues in the empirical chapters (Sections 4.5 and 5.9) where I discuss the research and its conduct, and articulate the procedures to detect and mitigate potential bias or limitations. My positionality in the AR continuum can be stated, therefore, as of an insider to RAPID-MIX, with the goals, values, and background knowledge described above, conducting an action research study and collaborating with other insiders who are engaged and equally committed to the success of the project and interventions under study.

Understand the problem

— George Pólya (1945)

Chapter 4

First Action Research Cycle: Preliminary Research and Action Learning

This chapter examines the primary research activities, findings and insights of the first action research cycle. This cycle encompasses preliminary research and interventions for the user-centred design process of RAPID-MIX, including the definition and production of a methods framework, preparation and deployment of a set of user-centred design interventions, collaboration in the assessment of the first generation of prototypes, and preparation and deployment of a set of user-centred design interventions. Section 4.1 reports on the results of an initial diagnosis which addressed both the preliminary goals of the RAPID-MIX consortium and problems which required immediate investigation and action at the time. In Section 4.2, I introduce the *User-Centred Design Actions* framework, a methods framework I developed in collaboration with other stakeholders in RAPID-MIX. In Sections 4.3 and 4.4, I report on early UCD actions, including scheduling and general planning, procedure and results. In the latter study, I employed a subset of RAPID-MIX technologies and focused on the design and implementation of software tools for prototyping with interactive machine learning in a research context. In Section 4.6, I reflect on the findings from the first action research cycle and on the insights that originated from UCD actions and software studies in relation to my research problem and research questions.

4.1 Introduction to the First Action Research cycle

I defined the first action research cycle as including the events and aspects of the initial period of RAPID-MIX of most significe to the preparation for development of RAPID-MIX API. As the reader can observe in Figure 4.2 (p. 79), I delimited the first action research cycle, as starting with my entrance to the project and ending at the period encompassing preparatory activities leading to the second round of concurrent prototyping and evaluation—the first 15 months of RAPID-MIX, approximately. This first action research cycle included the first milestone of RAPID-MIX (milestone M1), "Deployment in relevant environments of the RAPID-MIX proofs-of-concept", and part of the second milestone (M2), which produced the second generation of prototypes. In RAPID-MIX, these milestones included different aspects and activities—including evaluation, project data management, project sustainability, technology integration, rapid prototyping, user-centred design, etc.—which interacted through different work packages (see figure 1.2).

Of interest to the scope of my research are the activities in which I collaborated within the RAPID-MIX consortium. These activities include the production of several deliverables for Work Package 2 (WP2) within the period leading to the first milestone. First, I designed collaboratively the *User-Centred Design Actions* framework, a methods framework that was the core contribution of *D2.1 UCD Methodology* (Bernardo et al., 2015), the first deliverable which the RAPID-MIX consortium produced for WP2. Afterwards, I collaborated in the production of *D2.2 Design Guidelines for Prototyping* (Bevilacqua et al., 2015) to extend the *UCD Actions* framework with the first guidelines for prototyping. I made this contribution based on the work and insights I obtained from deploying the first set of UCD actions and first generation of prototypes.

Diagnosis

Action Research literature (Kock, 2007; Susman and Evered, 1978) recommends an initial stage of diagnosis in an action research cycle, where the researcher identifies a problem to be solved or an opportunity for improvement at the client organisation. Following these recommendations, I conducted an independent diagnosis of the organisational situation and goals of RAPID-MIX to understand what would best relate to my unit of analysis and that I could improve in the project with my intervention. Given the specific circumstances of my entry to RAPID-MIX project—i.e. after the preparatory work accomplished at grant writing time and project funding approval—the first diagnosis effort consisted of acquainting myself with the different aspects of the project.

This first effort entailed reading thoroughly and analysing the RAPID-MIX bid document to the funding body, the European Commission. The bid document contained an in-depth contextualisation and description of the project, exposing project goals, structure of the process and work plan to take over. The bid document also provided, additionally, a broad description of the organisational and technological capabilities of each of the consortium's institutions—including team members, production capacity, individual team members, and technology portfolio (Jordà et al., 2014). These have been briefly described in my overview of RAPID-MIX project goals, stakeholders and work plan, in Section 1.2. Section 1.2.2 provided a summary of the initial innovation portfolio and prior technological advances in RAPID-MIX.

In the list of threats and risk-mitigation measures of the RAPID-MIX bid document—i.e. Section

3.2.4, *Analysis of risks* (Jordà et al., 2014, p. 60)—there were identified threats related to user-centred design and user engagement:

> - *Weak foundation in User-Centred Design (inconsistent methodology/insufficient subjects) – To minimise this risk, special care [has been] taken to have a dedicated [work package] task devoted to design the guidelines for design which will be iteratively tested in the UCD sessions. As for user recruitment, there should be considerable access to potential users through the participation of UPF, GS and IRCAM.*
> - *Not enough engagement of industrial users of the RAPID-[MIX] API or MIX products – [...] the RAPID-[MIX] API will be presented and tested in the Music Hack Days in order to widely disseminate our technologies and start creating a user community for our technologies, which will be able to directly interact with our technologies through the development platforms...*
>
> (Jordà et al., 2014, p.60)

I considered these threats noteworthy because they revealed that the RAPID-MIX consortium recognised the importance of design methodologies and user engagement. As previously mentioned in Section 1.2.3, the UCD methodology was originally intended to support the engagement of industry users and consumer end users. Table 4.1 presents the primary user categories that the RAPID-MIX consortium agreed on in this period of the project. At this initial stage, users considered in direct relation to RAPID-MIX were SME software developers and consortium stakeholders with a more technical role closer to development. Users of the SME MIX products were also included in this category. Users outside the consortium included people interested in and capable of subsuming disparate technologies towards their goals, projects and products, such as hackers[1] and makers[2].

The above mentioned threats also revealed the intention of the RAPID-MIX consortium to mitigate problems through the deployment of UCD methodologies. This included informing the development of the RAPID-MIX API with insights resulting from user engagement and parallel prototyping activities. Activities such as focus groups, workshops and participatory design activities would in turn inform other work packages and activities, such as agile prototyping, API development and evaluation. It also included the evaluation of the outcomes and success of the application of the UCD methodology. Given the responsibilities that were previously established for my role—i.e. focusing on WP2 and attending to the overall needs of the project, as I previously detailed in Section 1.2.4—this posed for me an area within RAPID-MIX providing a significant opportunity for improvement. Here I could intervene in the definition of the methodology and generate knowledge through my research about the application of UCD techniques.

[1] The application of the term "Hacker" here is without any pathological or criminal association, as it is usual in popular culture. Rather, hackers are technically-skilled individuals that are able to build and re-purpose computers, software and other non-trivial technologies.

[2] The Maker movement is a Do-It-Yourself culture typically associated to tinkering with electronics, bricolage, open-source hardware, and physical spaces for fabrication such as "fab labs" https://www.hackster.io/survey.

Table 4.1: Initial RAPID-MIX user categorisation

Relation to the project	API users	MIX product end users
Directly linked to RAPID-MIX	RAPID-MIX SME developers (and other stakeholders)	End users of RAPID-MIX SME products
External to RAPID-MIX	Hackers, makers, other industry developers beyond RAPID-MIX SMEs	End users of other products built on RAPID-MIX API
	Some users—especially hackers, makers and students—making technologies primarily for themselves	

At this time, shortly after commencement of the project, the RAPID-MIX academic stakeholders conducted an initial survey about the organisational aspects of the RAPID-MIX SME stakeholders. This survey provided me with a situational perspective of the SMEs' a) technical capabilities concerning hardware and software production, b) workflows and methodologies, c) technologies used in their development processes and products, and d) their interests and needs concerning potential products that could leverage on technologies of the initial RAPID-MIX portfolio.

The survey was conducted in isolation, where each SME responded individually. As such it presented some limitations, principally regarding the big picture of the project, which was not unusual for stakeholders at that initial stage. For instance, the survey did not inquire about potential "knowledge spillovers" (Braunerhjelm et al., 2017) between stakeholders. Neither did it take into account a more comprehensive knowledge about how potential technology providers and recipients in the RAPID-MIX consortium could interact, or provide to each other holistically. This observation led me to recognise a potential opportunity to influence this engagement and foster knowledge spillovers between RAPID-MIX stakeholders early in the RAPID-MIX project.

Another opportunity for improvement in RAPID-MIX concomitant to my research, was related to the assessment of candidate technologies from the joint portfolio of technologies held by the consortium, with potential for prototypes that could inform and integrate the RAPID-MIX API. This opportunity involved performing a technical and hands-on exploration, where I would initially focus on the subset of Goldsmiths' technologies (i.e. Wekinator, Maximilian and GVF), and gradually expand the scope of exploration, to cover other partners' technologies—e.g. PLUX's BITalino and OpenSignals, MTG/UPF's Repovizz and Freesound, and ROLI JUCE.

Based on these considerations, I formulated my research questions for the first AR cycle as follows:

1. How might we support a pragmatic and effective approach to User-Centred Design in RAPIX-MIX that could also benefit my research?

2. Who are the stakeholders and the potential users of RAPID-MIX API and what are their needs, goals and values?

3. What is the space of compelling applications that RAPID-MIX technologies can provide for both RAPID-MIX consortium stakeholders and potential end users?

The following section introduces and examines in detail a methods framework which I have designed in collaboration with other RAPID-MIX consortium members. This methods framework aimed to support the necessary research to answer the first action research questions and my general research questions. Particularly, I intended it to guide the practice of UCD and data collection in RAPID-MIX, and support the identification and characterisation of the target user groups and possible contexts of use, and compelling applications for the development of prototypes of the RAPID-MIX API.

4.2 RAPID-MIX User-Centred Design Actions framework

The RAPID-MIX *User-Centred Design Actions* is a methods framework which aligns the cycles of User-Centred Design and Action Research with the goals and activities of the research-client project, and maps specific UCD techniques into actions for positive change. UCD techniques can be understood as tools to answer and refine key questions throughout the life cycle of a project. A *UCD Action* is a methodological construct which places emphasis on leading research through questions, intervention and reflection. It can be understood as an intervention where a UCD technique is deployed to answer questions about design, collect data for subsequent analysis and distil design knowledge.

The UCD Actions framework aims to provide methodological support for a dual and iterative process (Figure 4.1). This process subsumes and reconciles the stages of an UCD cycle (i.e. *Understand Users*, *Ideate*, *Prototype*, *Evaluate*, *Develop*), with the stages of AR—i.e. *Diagnosis*, *Action Planning*, *Intervention*, *Assessment*, *Reflection and Learning.* The UCD Actions framework comprises a set of stages and guidelines structured with leading questions and criteria:

1. Research–Client Agreement – subsumed from CAR and previously described in Section 3.4, and instantiated with my entry into the RAPID-MIX research-client system (Section 1.2.4).

2. Diagnosis ⇒ Identification of research questions to lead UCD actions – previously described in the context of CAR (Section 3.4) and instantiated for the first Action Research in (Section 4.1), and further elaborated in Section 4.2.1.

3. Action Planning ⇒ Selection of UCD techniques for application in interventions – further elaborated in Section 4.2.2.

4. Intervention ⇒ Deployment and report of UCD actions – further elaborated in Section 4.2.3.

5. Longitudinal and cross-sectional planning – elaborated in Section 4.2.4 and 4.2.5, respectively.

6. Assessment and Reflection – previously described in the context of CAR (Section 3.4).

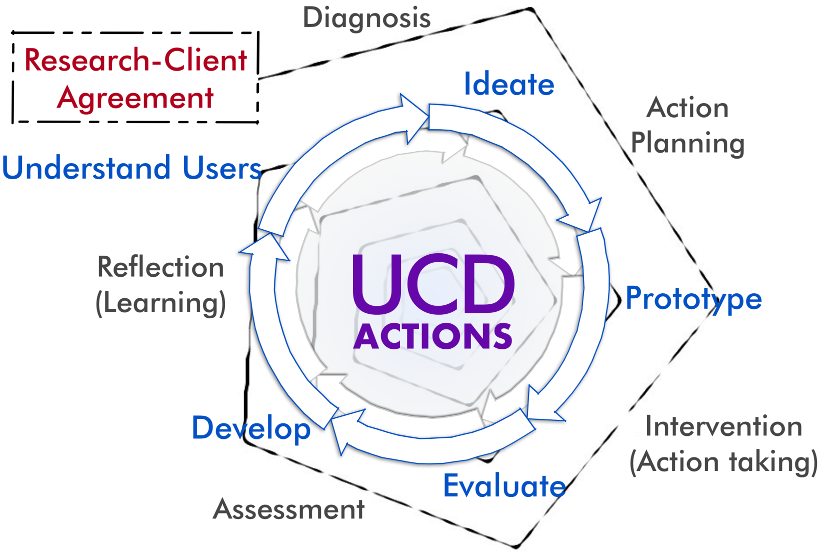


Figure 4.1: UCD Actions framework

The UCD Actions framework establishes an ethos for an actionable, collaborative and systematic approach to UCD throughout a project lifespan and its iterative cycles. On one hand, it supports UCD practitioners in the research-client project. On the other, it supports the academic action researchers working alongside the UCD practitioners as insiders, with the necessary planning, intervention and reflection for understanding the design of technological artefacts within the organisational contexts.

I designed the RAPID-MIX UCD Actions framework in collaboration with my PhD advisor Rebecca Fiebrink (who coined the term *UCD Action*). I also received input from other RAPID-MIX consortium members (Atau Tanaka, Frédéric Bevilacqua, Adam Parkinson and Sebastian Mealla). The RAPID-MIX UCD Actions framework was created primarily for internal guidance of the RAPID-MIX consortium (researchers and industrial stakeholders), and additionally, for other actors within the creative industries (e.g. professionals, academics, startups) and the general public.

With UCD actions, I aimed to create a methodological tool which could a) provide recommendations and structure to the design of interventions and collection of data in RAPID-MIX, b) support RAPID-MIX stakeholders in the deployment of UCD interventions and provision of practical insights, and simultaneously, c) support the necessary knowledge extraction and distillation for my research, during the longitudinal study of RAPID-MIX, for contribution to the research community.

The initial UCD Actions framework was published in official RAPID-MIX deliverables of WP2—*D2.1 UCD Methodology* (Bernardo et al., 2015), *D2.2. Design guidelines for prototyping* (Bevilacqua et al., 2015), and *D2.5 Final Specification* (Bernardo et al., 2017b)—the work package that encompasses all the user-centric aspects and activities of RAPID-MIX (see Figure 4.2). These documents were published online as a working manual available to the community and as part of

Table 4.3: Question frames and focus (Poggenpohl, 2000)

Question frame	Focus
What	Classification, specification
When	Time, sequence, context
Where	Location
Why	Reason, cause, purpose
How	Process, method, operation
Can	Possibility, probability
Will	Probability, trend
Do	Performance, action
Which	Comparison

the list of official resources of RAPID-MIX.

The RAPID-MIX UCD Actions framework was adopted as the working methodology of WP2 and applied by different RAPID-MIX stakeholders during the RAPID-MIX process cycle (see Section 1.2). Different UCD actions informed the work between the different work packages, and the general effort of bringing the background intellectual property and joint innovation portfolio (Section 1.2.2) from academic and industrial partners up to general industry standards, and to different businesses and user profiles to be able to use. In Sections 4.3, 4.4, 5.7 and 5.8, I examine some of the UCD actions in which I was directly involved.

In the following subsections, I present the constituent elements of the UCD Actions framework. I provide rationale into the different components and aspects of the UCD Actions framework, contextualising them with background knowledge in design research. I also describe how the UCD Actions framework prescribed for interventions to be planned, structured and conducted in a collaborative way, in order to support data collection and subsequent analysis.

4.2.1 Identification of research questions to lead UCD actions

The UCD Actions framework recommends interventions to be structured with leading questions. According to Davis (2015), framing the research question sets the stage for applying a research-driven design process to practical problem solving. Poggenpohl (2012) suggested the identification of valuable new knowledge as a starting point. She also identified different ways of asking a question, based on the frame and focus (Table 4.3).

Questions that could be asked and lead to a UCD action should be directed by the goals of understanding the characteristics of potential users and their experiences with developing technologies. They could be, for instance:

- What are the opportunities—recognised or unrecognised—for new technology to positively impact a given set of users in a given context?
- How is a particular design approach, if instantiated, likely to impact these users in a given context?
- How does a particular design instantiation impact these users in this context, in reality? What are the consequences for users' efficiency, effectiveness, satisfaction, ways of thinking and acting, relationships with each other, etc.? How do these change with time and experience, with context, or with the type of user?
- How do multiple design alternatives compare against these criteria of interest?
- Where are possible usability problems in a given design, and how might they be mitigated or corrected?
- What is the design space of possible technologies?

For instance, when a RAPID-MIX stakeholder had questions about next project steps that needed to be informed through interactions with users (of any type), the process of answering these could be understood as a *UCD action*. Through planning, structuring and deploying an intervention with users, e.g. a usability evaluation or other user engagement intervention, the RAPID-MIX stakeholder would harvest data for analysis and knowledge distillation to inform design.

4.2.2 Selection of UCD techniques for application in interventions

UCD actions in RAPID-MIX can be considered as methodological devices for operational research, which employ adequate UCD techniques for the specific set of circumstances which diagnosis unveils. The UCD Actions framework organises, shortlists and maps out UCD techniques according to specific factors which project stakeholders should consider at decision stage.

Appendix B presents a summary of techniques for user engagement and data collection with selection criteria: a) the purpose of the study, b) target user group, c) data output type, d) cost and e) project stage. Other factors which can also be included and considered for each case are: the number of team members involved in the UCD action, the target technologies, and the specific design goals.

For instance, a RAPID-MIX stakeholder needing to perform a usability evaluation, or any other user engagement intervention, would harvest data for analysis and subsequently distil information for design through a UCD action.

- Public or internal interventions, such as workshops, hackathons, and focus groups.
- SME site visits that include interviews, questionnaires, targeted development sections, and other activities with small numbers of developers using RAPID-MIX API.

- Periodic review of other RAPID-MIX API feedback from SME developers, e.g. via bug reports, feature requests, etc.
- Observation and interviews with SME personnel interacting with prototype technologies.
- Other actions that seek information from any type of users to inform the subsequent design of RAPID-MIX API and the SME MIX products.

4.2.3 Deployment and reporting of UCD actions

In the RAPID-MIX UCD Actions framework, we provided the following guiding questions to help identify how UCD actions in RAPID-MIX should be prepared:

1. What questions is the UCD action intended to answer?
2. Who will be the users engaged in this session?
3. What techniques will be used?
4. Who will run and document this action?
5. What activities and/or materials need to be prepared in advance?
6. What documentation will be collected?

Finally, I also provided recommendations in the UCD Actions framework for how UCD actions should be conducted on site, including ethical treatment of participants—following the European Code of Conduct for Research Integrity (All European Academies, 2017)—use of informed consent forms, and adequate documentation and data management techniques.

4.2.4 Longitudinal planning in RAPID-MIX

The RAPID-MIX UCD Actions framework calls for the necessity of planning interventions both longitudinally and cross-sectionally—i.e. both across the RAPID-MIX project life cycle and at a specific point in time, respectively. There were several reasons for this. Firstly, I aimed to make the UCD Actions frameworks adhere to the action planning stage of Action Research, and secondly, because the purpose of a UCD action was to be aligned with the general goals of RAPID-MIX. This included alignment with the design road map of RAPID-MIX API and of the RAPID-MIX industrial stakeholders. Finally, I wanted to ensure that the leading questions and techniques chosen for a specific UCD action would converge with the practical needs and issues of a specific deployment scenario, at a given time.

Figure 4.2 depicts the initial structure proposed within the RAPID-MIX UCD framework to frame and deploy UCD actions across the RAPID-MIX project life cycle. The diagram shows how UCD and Action Research cycles were framed in relation to the unit of analysis of my research, the

Table 4.5: Longitudinal planning of UCD actions

AR stage	AR cycle 1	AR cycle 2	AR cycle 3
Diagnosis	Identify SME partners' technologies and needs; Identify status of candidate technologies	Identify SME partners' technologies and needs; Identify status of candidate technologies; Use early prototypes to identify high priority API features	Identify status of candidate technologies; Identify high priority API features; Identify SME product goals (based on their assessment of market dynamics and capabilities of the RAPID-MIX API
Action Planning	Survey for assessing SME's technical standards and target MIX products; Prepare a co-design workshop for SME partners; Prepare activities for end-users during the Barcelona MHD; Prepare assessment tools;	Prepare local interventions at SMEs; Prepare local interventions at SMEs; Prepare demonstrators of RAPID-MIX API; Prepare assessment tools; Prepare co-design workshop;	Prepare evaluation of RAPID-MIX API; Prepare local interventions at SMEs
Action Taking	Demonstration of early prototypes (for industrial and end-users); Co-design workshop (for industrial-users, represented by SMEs); Involvement in the Barcelona MHD	Co-design workshop (for SME developers); Integration of feature sets, API prototyping; On-site interviews, local surveys, logs and feature requests from developers using early versions of the API; Demonstration of RAPID-MIX API Beta (for industrial and end-users)	Co-design workshop (for SME stakeholders and developers); Integration of feature sets, API refinement; On-site interviews, local surveys, logs and feature requests from developers using early versions of the API; Demonstration of RAPID-MIX API Release (for industrial and end users); Benchmarking and validation of MIX interfaces with end users
Evaluation	Review of UCD actions outcomes	Review of UCD actions outcomes	MIX product evaluation; RAPID-MIX API evaluation; General process evaluation
Specifying Learning	Inform Product design loop (WP2+WP4+WP5) Inform Evaluation loop (WP6)	Inform the selection of core functionalities and interface of RAPID-MIX API	Inform Evaluation loop (WP6)

RAPID-MIX API. I reconciled the cycles with elements in the RAPID-MIX project life cycle (e.g. milestones, deliverables of WP2), and artefacts such as the generations of prototypes, the RAPID-MIX API, and the MIX products, which had been planned for the project.

This diagram provided an early template for situating the different UCD actions that were planned and deployed in the project lifetime according to the three action research cycles. Table 4.5 presents the more detailed plan, which I proposed to the consortium as part of *D2.1 UCD Methodology.* In this preliminary plan, I defined the goals and intended UCD actions for the three years of the project. As the reader will notice in Chapters 5.2 and 5, there was a deviation from this plan in terms of the number of AR cycles that were deployed, two instead of three. This will be discussed in Section 7.4.1.2.

4.2.5 Cross-sectional planning for the first action research cycle

For the planning of early UCD actions, I pursued a set of goals which were informed by the research questions that I identified at the diagnosis stage (Section 4.1). Specifically, these goals included:

1. Assessing and aligning the consortium partners' thinking through ideation, design challenges and identification of scenarios.
2. Defining an initial set of target users and contexts of use, and evolving gradually to a more comprehensive definition.
3. Gathering direct insights about RAPID-MIX background technologies—i.e. using technology probes based on early proofs-of-concept and prototypes which integrated these technologies—from the future users of RAPID-MIX API to inform the production of design guidelines for RAPID-MIX API.

I participated in a consortium meeting hosted by IRCAM (Paris, May 20-22th, 2015) where we agreed on, and began planning the early UCD actions for the first milestone. These UCD actions included:

- a co-design workshop (UPF, Barcelona, June 16th, 2015)—I collaborated in the design and deployment of this UCD action with RAPID-MIX stakeholders and SME developers as industrial users, to understand the perceived space of possible applications, end users and contexts of use of candidate technologies, and to inform the design of prototypes and of RAPID-MIX API (Section 4.3);
- a 24hr-hackathon at Music Hack Day (Sónar+D, Barcelona, June 18-19th, 2015)—I collaborated in different aspects of this UCD action, which was lead by MTG/UPF and involved RAPID-MIX stakeholders as tech providers of the RAPID-MIX challenge at Sónar+D. This action was preceded by a pre-event workshop (Hangar, Barcelona, June 17th, 2015) which involved hackers in technology induction sessions and a hands-on session with candidate

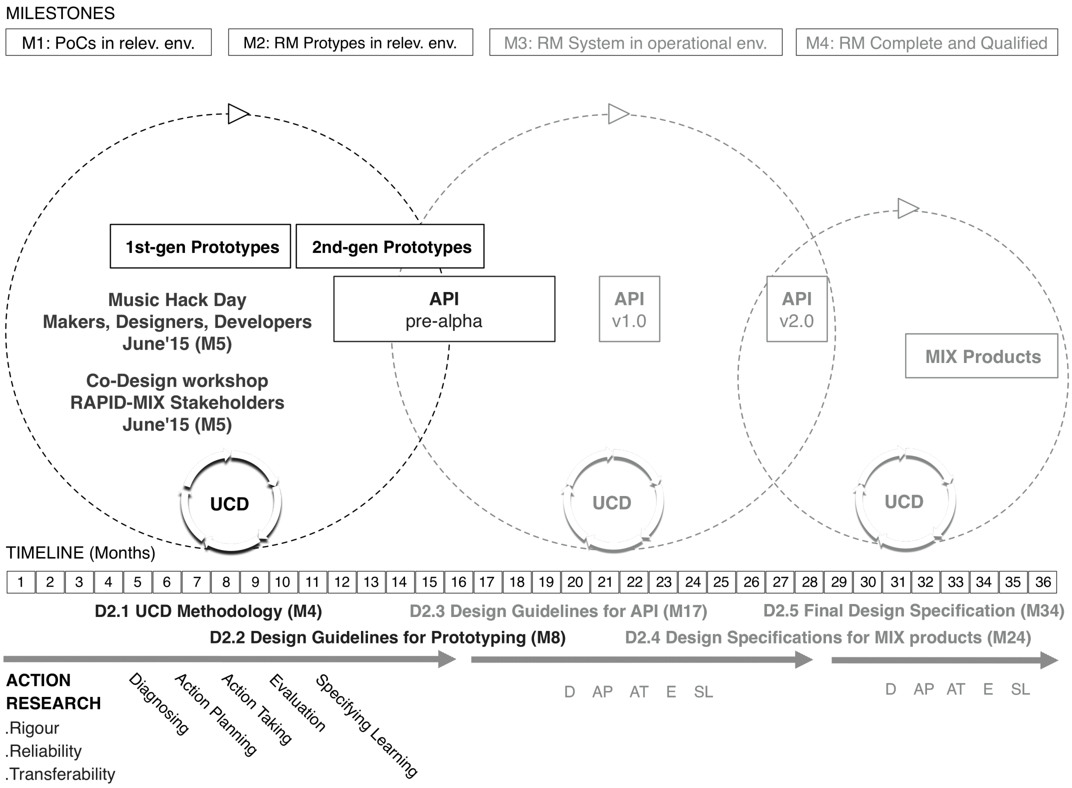


Figure 4.2: First Action Research cycle UCD actions

RAPID-MIX technologies demonstrators as technology probes. Both events shared the goal of engaging with hackers as end users to obtain user insights and uncover user needs and goals with RAPID-MIX technologies. This UCD action will be examined in Section 4.4.

Given the set of UCD actions agreed on within the RAPID-MIX consortium and my own initial diagnosis, I engaged in collaboration for their preparation and more detailed planning. This involved considering alternatives concerning the selection of effective UCD techniques for application with RAPID-MIX stakeholders and potential end users. In the following sections, I examine how the different UCD actions in which I was involved were planned and deployed, detailing for each one of them, my role, the adopted procedure, results and discussion of outcomes.

4.3 Co-Design workshop with RAPID-MIX consortium stakeholders

The Barcelona co-design workshop (UPF, Barcelona, June 16th, 2015) was the first of a series of early UCD actions in which I employed brainstorming and ideation with RAPID-MIX internal stakeholders about future RAPID-MIX technologies. I aimed for the co-design workshop to help the RAPID-MIX consortium understand the space of possible innovative applications and products that could be built with RAPID-MIX technologies, and inform the design of early prototypes and

(a) (b)

Figure 4.3: Co-design workshop with RAPID-MIX stakeholders. a) Group engaged in the ideation process during the application of the bootlegging technique. b) Group presenting their final idea, *Le Cook Rapide*, using a storyboard.

of the RAPID-MIX API. I split this UCD action into two brainstorming and ideation sessions. In the morning session, we focused on open-ended, divergent design thinking (Brown, 2009). In the afternoon session, we applied convergent design thinking around previously identified themes of interest.

I collaborated with my PhD advisor, Rebecca Fiebrink, and Atau Tanaka (both from the Goldsmiths, University of London team) and Sebastian Mealla (MTG/UPF) in the design and deployment of this UCD action. I also designed the research materials and internal communication of the intervention (see Appendix C). These materials included a design brief with the workshop program, two sets of ideation cards, templates for storyboards and design scenarios, and consent forms, and one section of the RAPID-MIX institutional website.

4.3.1 Divergent design session procedure

My main goal for the co-design workshop morning session was to mix participants coming from different institutions within the RAPID-MIX consortium and with different backgrounds and roles, and have them generate as many ideas as possible for new applications in the broad areas related to RAPID-MIX technologies. I intended to foster "knowledge spillovers" (Braunerhjelm et al., 2017) between participants, and have them think creatively and come up with ideas about application with the types of interaction and features supported by RAPID-MIX technologies. However, I aimed for participants to do so without being constrained or overly attached to any specific RAPID-MIX technology. I was looking for ideas in areas that encompassed RAPID-MIX technologies—for instance, machine learning, biosignals and data repositories—to ground the generation of new concepts and applications, and new combinations of users, contexts of use, and technologies.

I proposed to use a variation of a multidisciplinary brainstorming technique called *bootlegging* (Holmquist, 2008) as the main UCD technique of the session, aiming for an open-ended, divergent-thinking morning session. Holmquist defined *bootlegging* as "a brainstorming technique that can be used to generate ideas in situations where the problem area or technology is fairly well defined but still open for new application ideas. It is particularly suited to situations where participants come from many different disciplines." (Holmquist, 2008, p. 158). Firstly, bootlegging appeared to be a suitable technique for the particular goals of this UCD action, particularly, brainstorming and ideation of new applications for RAPID-MIX technologies. Secondly, time constraints are strictly enforced in bootlegging, which allowed it to be deployed in a half-day session and therefore appropriate to the time constraints of the co-design workshop. Third, in my literature survey of UCD (Section 2.2.2), I identified that bootlegging had been previously employed at Goldsmiths (Correia and Tanaka, 2014), which gave me additional assurance about its efficacy. Finally, bootlegging was also accepted by the other collaborators as the main UCD technique to apply in the UCD action.

Additionally, I employed two other techniques in the divergent design thinking session to make brainstorming more efficient and effective. This included preparing in advance a set of ideation cards and a storyboard template as material for participants to use. I designed these research materials specifically for the workshop and employed RAPID-MIX branding to give them enhanced credibility (Figure C.1, Section C.3, Appendix C). My intention in using the ideation cards as a UCD technique was similar to that of Halskov and Dalsgård (2006). Halskov and Dalsgård employed inspiration cards in a workshop as a collaborative method to create new concepts for design, through the combination of findings from domain studies with inspirational applications of technology.

I intended the ideation cards to inspire and stimulate participants. I wanted participants to articulate their knowledge and practices in their technological domains within their groups and in the shared social context of the workshop. I also used the ideation cards to provide a constraint for technological domains related to RAPID-MIX. Additionally, I introduced blank technology cards for participants to fill out with any relevant suggestion that they would see as missing. The ideation cards introduced the domains listed below, which grouped into the following categories:

- *Interaction Modes* – e.g. motion sensing, biosignals, tangible interaction, multitouch, gestural interaction, haptics.
- *Software Libraries* – e.g. affective computing libraries, signal processing libraries.
- *Applications* – e.g. interactive machine learning applications, multimodal data repositories, sound databases.

The elementary steps of the morning session were listed in the workshop program (Section C.2.2, Appendix C). Sixteen members (14 males, 1 female) from all the RAPID-MIX member institutions participated in the session. Atau Tanaka and I acted as session moderators and time keepers. Additionally, since I was in charge of documentation, I also set up the cameras to record video of the session. As the session began, I distributed a document with the design brief, program and groups (Appendix C), and a consent form to each participant. Atau Tanaka debriefed the participants on the goals of the session and the activities and materials to be used.

We began with the *generation* stage of bootlegging at a fast pace, in which we asked participants to write as many instances of potential users of RAPID-MIX technologies as possible in yellow post-its (10 minutes). After generating *user* post-its, participants were instructed to post them on the back wall. Afterwards, participants were asked to generate instances of potential contexts of use for RAPID-MIX technologies, using pink post-its (10 minutes). After generating *context of use* post-its, participants were instructed to post them on the back wall. Atau Tanaka and I made sure, as time enforcers, that participants completed the task in timely fashion and adhered rigorously to the schedule, in order to keep the fast pace required for bootlegging.

Next, we divided the sixteen participants into four breakout groups (see Section C.2.3, Appendix C). We organised groups at planning time to be heterogeneous in terms of background, profile and RAPID-MIX partner institution. Each group was also assigned a coordinator chosen from the RAPID-MIX research institutions—Frederic Bevilacqua (IRCAM), Norbert Schnell (IRCAM), Rebecca Fiebrink (Goldsmiths) and Sebastian Mealla (MTG/UPF). We also provided the additional materials to each group, which included the set of ideation cards and the storyboard template.

We employed the *mixing* and *brainstorming* stages of *bootlegging* (Holmquist, 2008). We communicated the goal as having participants come up ideas that would be mixed to create random combinations of User, Context of Use, and Technology, in several iteration rounds. We instructed group participants to build up these combinations by posting one *user* post-it and one *context of use* post-it (randomly selected from the wall) into a technology ideation card (10 minutes). Participants of each group collected both *user* and *context of use* post-its from the back wall and used them to build combinations at their desk (Figure 4.3a). Atau Tanaka then instructed participants to provisionally switch work tables to shuffle-up the other groups' combinations—i.e. shuffle the *user* or *context of use* post-its associated to an inspiration card. Next, we instructed participants in each group to go back to their tables and select four combinations and brainstorm them (40 minutes). Three groups of participants pre-selected four combinations, and one group pre-selected only three combinations.

Finally, we employed the *final idea* stage of bootlegging. We instructed participants to pick one of the four combinations they had previously brainstormed, and describe it using a storyboard. We also asked participants of each group to select a spokesperson to deliver a presentation to the whole session group explaining the final idea on the storyboard (40 minutes). The spokesperson from each group delivered a presentation, using the storyboard to explain the final idea (Figure 4.3b). All presentations were video-recorded and archived in the RAPID-MIX video repository. All post-its, combinations, storyboards were documented and archived.

4.3.2 Results

The divergent session of the co-design workshop yielded the following data:

a) 103 *user* post-its, containing one instance of a potential user of RAPID-MIX technologies,

b) 150 *context of use* post-its, containing one instance of a potential context of use for RAPID-MIX technologies,

c) 52 combinations of one ideation card, one *user* post-it and one *context of use* post-it – total number of combinations which included:

   i) 4 combinations selected as the final idea used to describe a scenario in a storyboard and in a live presentation to the group.

   ii) 11 pre-selection combinations.

   iii) 5 invalid combinations (i.e. that were missing either a user post-it or context of use post-it).

d) 4 storyboards based on the final ideas of four groups, depicting potential scenarios for RAPID-MIX technologies, in terms of user, context of use, and technology.

e) Video recordings of the session and of the presentations of each group's storyboard with the final idea.

In the weeks following the UCD action, I informed RAPID-MIX stakeholders with a report. I tried to develop a better understanding of this data, in particular, of the final ideas and the users and contexts of use which participants proposed. I captured and complemented the final ideas with additional detail from the video recordings to communicate them more effectively. I also organised and classified the user and context of use data to provide a synthesis. These efforts are described in the following sections.

4.3.2.1 *Final Ideas and Storyboards*

In order to capture with more detail the final ideas which the four groups of participants generated, I first transcribed each group's storyboard (Figure C.3, Appendix C). I also interpreted the video recordings of each group's spokesperson presenting the final idea to complement the storyboard transcription. Additionally, I extracted a theme and a user story using a standardised format[3], "As a <type of user>, I want <some goal> so that <some reason>". The following subsections describe the outcomes of this synthesis.

Storyboard 1 – *Le Cook Rapide* (Figure C.3a)

Theme: Interactive culinary performance in a shared social environment

Group members:
Frederic Bevilacqua (Coordinator, IRCAM)
Panos Papiotis (MTG)

[3] User Stories, https://www.mountaingoatsoftware.com/agile/user-stories

Felix Faire (ROLI)

Xavier Boisserie (ORBE)

Ideation Card: Applications – Interactive Machine Learning

User post-it: Cook cuisine

Context post-it: Build something to impress your friends

Storyboard transcription:

1. *User chooses cookbook recipe.*
2. *Recipe is matched to Rapid-Mix recipe.*
3. *User makes recipe with augmented tools. It is recorded and provides feedback. Goes to 4.*
4. *User provides iterative feedback on the result. Goes to 3.*
5. *User shares data.*
6. *New users respond, give feedback and choose recipes.*
7. *Users remix and improve each other's results.*
8. *Internet and export performance stream.*

Observations from video footage:
Participants depicted *Le Cook Rapide* as a scenario creatively built upon popular TV cooking shows, where end users' goal is setting up a rapid, interactive audiovisual culinary performance, to show off their culinary skills and impress their friends. Participants conceptualised an interactive machine learning workflow in an augmented cooking application. The iterative loop of an IML workflow, which is described in steps 3 and 4 of the storyboard, is applied to building a model of the cook's embodied interaction with the augmented cooking utensils while cooking a specific recipe. Cooks provide feedback to train and improve the model. The resulting model and data, and the associated recipe end up populating a recipe collection, which is improved through community and collaborative feedback.

User story:
Based on the analysis of this storyboard and the participants' presentation, one could formulate a possible use story as: "*As a cook, I want to record an augmented culinary performance so that I can impress my friends with my culinary skills*".

Storyboard 2 – *Bored Architects* (Figure C.3b)

Theme: Mobile collage app for conferences

Group members:
Norbert Schnell (Coordinator, IRCAM)
Carles F. Julià (MTG)
Jack Armitage (ROLI)
Tomek Jarolim (ORBE)

Ideation card: Near-Field Communication
User post-it: Architects
Context post-it: Being bored

Storyboard transcription:

1. *Architects (highly specialised professionals) at a boring event (congress, dinner, ...).*
2. *One of them starts the game.*
3. *He chooses the media (sound, tech, image, camera, microphone, touch gesture).*
4. *. . . and sends it to someone nearby (who he [does/does not know].*
5. *The receiver receives a message/link/invitation to continue BA collage.*
6. *He/she adds a media while being [displayed] a fragment of the received media.*
7. *After a number of iterations, [each] participant receives a message/link/invitation to unfold. . .*
8. *. . . the resulting collage. Voila!*

Observations from video footage:
Participants took advantage of the "empty" ideation cards and managed to push through an idea comprising the Near-Field Communication (NFC) technology, which was not relevant to the RAPID-MIX technology portfolio. Interestingly, there was also a lack of other prominent functionalities related to the RAPID-MIX technology portfolio. Participants depicted *Bored Architects* as a scenario associated with a specialised professional sector, architecture. The context of use was in a professional conference where participants were bored, and some of them would decide to engage in a creative mobile application that worked on co-located networks supported by NFC. The proposed application concept was a digital, collaborative and mobile version of the traditional *cadavre exquis* game, that concatenated different kinds of media in a creative thread of contributions.

User story:
A possible user story based on this storyboard could be: "*As a bored architect in a conference, I want to engage with co-located fellow architects so that I can add creative input into a mobile app thread about the conference*".

Storyboard 3 – *Intelilight* (Figure C.3c)

Theme: Smart ambient display of patient status for hospital wards

Group members:
Sebastián Mealla (Coordinator, MTG/UPF)
Mick Grierson (GS)
Andrés Bucci (RS)
Hugo Silva (PLUX)

Idea card: Applications – Multimodal data repositories
User post-it: Doctors
Context post-it: Light Visualization

Storyboard transcription:

1. *Doctor is a in a ward monitoring data from many patients.*
2. *The patients are [in] their beds.*
3. *[Each patient's] data is streamed to a cloud service and stored in the repository.*
4. *Using machine learning the data is classified and compared.*
5. *The doctor receives a verification about a patient's state that requires his intervention.*
6. (empty)
7. *The lighting in the room changes to affect the patient's well-being.*

Observations from video footage:
Participants depicted *Intelilight* as a scenario in the healthcare domain, where a physician uses an ambient display application to learn about the health status of his patients in a specific ward at one glance. Participants conceptualised the support system, *Intelilight*, as a composition of prominent features of the RAPID-MIX technology portfolio. Notable elements in this scenario were the use of machine learning to support the smart application which provides non-disruptive and gradation feedback to help the physician recognise the need for an intervention. The learning of patients' conditions would be achieved through the classification and comparison of real-time biosignals. Also interesting was the conceptualisation of where the "learning" happens. Given that the multimodal data repository is a central component of the cloud service infrastructure, server-side machine learning provides for the "smartness" of a local system either through notifications, or through the deployment of a local machine learning model.

User story:
A possible user story based on this storyboard could be: "*As a doctor, I want to use Intelilight so*

that I can be aware if a patient needs intervention at a glance".

4.3.2.1.1 Storyboard 4 – *Guitar 911* (Figure C.3d)

Theme: Smart learning environment for virtual reality with haptic feedback.

Group members:
Rebecca Fiebrink (Coordinator, GS)
Sergi Jordà (MTG)
Emmanuel Flety (IRCAM)
Francisco Bernardo (GS)

Ideation card: Multimodal data repository
User post-it: Humanoid Robot
Context post-it: Learning a practical skill

Storyboard transcription:

1. *The guy wants to impress his love with a guitar performance.*
2. *Database of multimodal data (sensor, audio, video, score,. . .) from guitar players.*
3. *Guy has haptic glove. . .*
4. *And VR headset w/ 3D avatar guitar player*
5. *His performance/practice is measured*
6. *Info for DB is used to evaluate and instruct.*
7. *He learns guitar and impresses the girl.*
8. *She learns the drums*

Observations from video footage:
Participants depicted *Guitar 911* as a scenario where an end user aims to accelerate learning of guitar technique to impress his girlfriend. The user resorts to a virtual reality (VR) learning environment with gamification. Participants conceptualised a composition of functionalities provided by different technologies in the RAPID-MIX portfolio. One central element is a multimodal database as part of the applicational infrastructure which stores all the required data modalities of the apparatus. There is a sensor and actuator—i.e. the hand exoskeleton—that captures the end-user's embodied interaction data and provides haptic feedback for accelerated learning of guitar playing technique. Notably, there is also a machine learning component in the system, which learns (a) procedural model(s) of the end-user's guitar performance for evaluation and direction of the system's feedback and tuition.

User story:

A possible user story based on this storyboard could be: "*As a newbie guitarist, I want to use Guitar 911 so that I can learn to play a serenade and impress my girlfriend*".

4.3.2.2 *Users and Contexts of Use*

In order to develop a better understanding of the large quantity of post-its that participants generated in the divergent design session, I employed affinity diagramming to help sensemaking and interpretation of the data. I gathered all the post-its and compiled the instances of users and context of use from the post-its (see Figure C.4 and Figure C.5 in Appendix C, respectively) into a spreadsheet.

Next, I organised and sorted these instances of users into an initial set of categories. Given that participants described user instances mostly as a role or activity, I used broad areas of activity which could aggregate the user instances. Namely, this set of categories included: *Developers*, *Education*, *Music*, *Gaming*, *Healthcare*, *Academia*, *Agriculture*, *Arts*, *Industry*, *Entertainment*, and *Other*.

I did two additional rounds of prioritisation and aggregation of the resulting categories. I discuss next the resulting clusters and how they relate to their content:

- *Developers* – 'developers' (and related profiles such as 'hackers' or 'coders') had a significant presence in the list of user instances. This 'developer' appeared juxtaposed with attributes such as 'new' or 'creative' or 'learning' in the list of user instances. This directs the attention to user-developers, in addition to professional developers, in development activities outside their professional context.

- *Education & Academia* - I extracted this category from a heavily populated cluster which included user instances such as 'teachers', 'students' and 'researchers'.

- *Healthcare & Sports* – I extracted this category from a heavily populated cluster which included user instances related to different sports (e.g. 'athletes') and to professions in healthcare and physiotherapy.

- *Art, Music, Gaming & Entertainment* – this super group aggregated different kinds of artists and roles related to artistic production. One set of user instances which I found particular significance in was associated to the term 'musical instrument builders'. There were a number of different associated terms such as 'luthiers', 'new instrument designers', 'plugin developers'.

- *Industry, Automation & Monitoring* – another super group organised more professional roles (e.g. 'product designers', 'startups'), with robotics and automation (e.g. 'robots', 'drone operator', 'buildings', 'plants', 'environment'), and with monitoring, surveillance and security forces (e.g. 'police', 'security').

One other high-level categorisation that I found significant was a binary classification of users according to:

- *Proxy users* – people who employ their judgement with reference to end users, or that role-play end users in requirement elicitation activities (Preece et al., 2015). On one hand, this category can refer to developers as users of RAPID-MIX technologies as tools, and as creators of products for domain-specific end users. On the other, it can refer to internal users of RAPID-MIX technologies that may have a more ambiguous role in the producer-consumer continuum, such as researchers' building tools, teacher applying pedagogical tools, domain experts using domain-specific tools, or SMEs stakeholders undertaking product management. This category aggregates users from the previously defined categories in *Developers*, *Education & Academia*, and *Industry.*

- *End users* – people who use the products created with RAPID-MIX technologies in domain-specific activities and tasks.

I organised and clustered instances of *contexts of use.* This provided a similar set of categories related to the user categories, as follows: *Artistic*, *Commercial/Retail*, *Development*, *Educational*, *Gaming*, *Healthcare*, *Intimate/Familiar*, *Social*, *Musical*, *Political*, *Recreational*, *Touristic*, *Sports*, *Social*, *Transportation*, *Work*, and *Other.* I also identified a set of themes which shows the breadth of affordances of RAPID-MIX technologies as perceived RAPID-MIX stakeholders. These themes show as contrasting relationships, which include:

- *Making vs. Using* – this theme shows the perception of stakeholders regarding the potential of RAPID-MIX technologies for integration in development tools (for both rapid prototyping and production tools) and end-user products. This theme also appears to relate to the dichotomy of *Proxy user/End user*, the binary classification of instances of user. *Proxy Users* seems to relate to clusters of contexts of use related to *Making*—i.e. creation and production activities, e.g. *Development*, *Product Development*, *Rapid Prototyping*, *Exploratory Design. End Users* relates more to the contexts of use clustered under *Using*, that is, the use of a product.

- *Work vs. Recreational* – this theme shows how RAPID-MIX stakeholders perceived the potential of RAPID-MIX technologies for application in both professional and work settings, and recreational contexts and activities. Work contexts of use included, for instance, *In Conferences*, *Meeting*, *In the office*, *In the factory.* Recreational contexts included physical exercise and leisure contexts such *Jogging*, *Gym*, *Cooking*, *At the beach*, and *Going to a concert.*

- *Public/Social vs. Personal/Intimate* – this theme highlights how RAPID-MIX stakeholders perceived the variety of contexts-of-use concerning social modes of application and use of RAPID-MIX technologies. Public/Social contexts of use include, for instance, *Dancing*, *In* *a meeting*, and *Making music together*. *Personal/Intimate* includes for instance *Meditating*, *Having Sex* and *Sleeping*.

4.3.3 Convergent design session procedure

My main goal for the convergent design session of the co-design workshop was to have participants converge upon ideas for building prototypes with RAPID-MIX technologies which could, in turn, inform the design of MIX products and of RAPID-MIX API. I wanted to have the same participants of the divergent design session engaging in this activity, immediately after the earlier brainstorming on potential users and contexts of use for the broad areas of RAPID-MIX technologies. As with the morning session, for the convergent design session of the co-design workshop, I wanted to have different RAPID-MIX stakeholders working together, exposing their goals and needs, as well as their common and competing interests.

However, while the morning session was intended to be more abstract, open-ended, and playful, the afternoon session was intended to have participants build scenarios around prototypes of concrete applications of technologies from the RAPID-MIX portfolio. Scenario-based design is one of the most used UCD methods (Vredenburg et al., 2002), where stakeholders of a design process build a brief story involving users as characters in order to express a design problem or research questions.

In the convergent design session I employed scenario building for having participants propose prototypes that integrated RAPID-MIX technologies and identify the learning that these prototypes would provide. By eliciting scenarios and prototypes from RAPID-MIX stakeholders, I wanted to learn about their vision, in terms of concepts and motivation, and their goals for RAPID-MIX technologies. I wanted to obtain from RAPID-MIX stakeholders a high-level specification and design rationale for useful prototypes, including relevant functionalities and issues, and goals for supporting end users of RAPID-MIX technologies.

In order to assist these goals, I prepared in advance a set of supporting research materials to be used in the convergent design session. In addition to the design brief and detailed program (Section C.2.2, Appendix C) that was presented in the previous section, these materials included the *Technologies* section of the RAPID-MIX institutional website, a second set of ideation cards, and a specific document template for capturing the scenarios built by participants; these are each described below.

I designed the RAPID-MIX institutional website[4] in collaboration with designer Myah Grierson, and with direct input from other Goldsmiths' team members (Atau Tanaka, Rebecca Fiebrink, Mick Grierson). It was completed, published and communicated to the internal mailing list of the RAPID-MIX project in the week before the workshop.

I also designed a set of ideation cards to be used in the convergent design session, to complement the website with a more local and tangible session-supporting material. These cards were intended

[4] RAPID-MIX institutional website, http://rapidmix.goldsmithsdigital.com

to provide design constraints to scenario building and to facilitate the discussion about future prototypes. Given the purpose of the convergent design session, these cards were more concrete and focused on specific RAPID-MIX technologies and technology providers (see Appendix C, Figure C.2).

Additionally, I designed a document template for participants to capture the scenario information from each group. I also designed these materials with the RAPID-MIX branding for enhanced credibility. The document template contained the following questions:

- User – Who is the user? What is the market space for the product?
- Technology – What technologies are involved? How are they combined?
- Prototype – What do we learn from this prototype? What benefits does it bring to the project's goals?

The participants of the convergent design session were the same who attended the divergent design session in the morning—the sixteen members (myself included) from all the RAPID-MIX member institutions. We began the convergent design session with Sebastian Mealla (MTG/UPF) presenting eight main ideas (for about 2 minutes each) which had been identified in a previous meeting in Paris, May 20-22, 2015, at IRCAM (see excerpt of the Paris meeting minutes in Section C.1, Appendix C). After Mealla presented these ideas as themes for inspiring the scenario building, we then established breakout groups through a sign-up process. We asked for participants to sign up to a group working on specific theme.

At this point there was a slight deviation of the proposed program. Some participants suggested reducing the number of themes and aggregating them. Instead of enacting the sign up process for two iterations of scenario building, the whole group engaged in a brief discussion (for about 10 minutes), where four encompassing themes were extracted from the initial eight themes and agreed on for the rest of the session. I wrote these themes on the whiteboard and participants, based on their interests, signed up to a group associated to one of the four themes, by writing next to the theme. The results of the sign up process were as follows:

- Biosignals repository and exploring databases
  - Panos Papiotis (MTG/UPF)
  - Hugo Silva (PLUX)
  - Sebastian Mealla (MTG/UPF)
  - Andrés Bucci (Reactable)
- Processing in hardware
  - Mick Grierson (Goldsmiths)
  - Emmanuel Flety (IRCAM)

  - Carles F. Julià (MTG/UPF)

- Designing expressive interactions
  - Sergi Jordà (MTG/UPF)
  - Frederic Bevilacqua (IRCAM)
  - Rebecca Fiebrink (Goldsmiths)
  - Francisco Bernardo (Goldsmiths)
- Web applications
  - Norbert Schnell (IRCAM)
  - Jack Armitagge (ROLI)
  - Felix Faire (ROLI)
  - Xavier Boisserie (ORBE)
  - Tomek Jarolim (ORBE)

After participants signed up and gathered around the work tables, I explained the rest of the procedure for one iteration, instead of the two originally planned (which took around 15 minutes). I challenged participants to come up with specific scenarios for application of RAPID-MIX technologies according to the agreed themes. Specifically, I explained that we wanted them to elaborate scenarios based on these themes, with the identification of target users and of the specific assortment of technologies from the RAPID-MIX portfolio. I also explained that we wanted to gather insights about what to build and how to build candidate prototypes for informing the design of MIX products and the RAPID-MIX API. At this point, I delivered the document templates and the ideation cards. I explained that to answer the questions "Who is the user?" and "What technologies are involved? How are they combined?" of the document template, participants should generate a set of instances of users, select a subset of RAPID-MIX technologies and then brainstorm them.

I stated that we wanted participants to come up with scenarios proposing prototypes that could provide useful insights about what to build for future RAPID-MIX technologies. I further explained that regarding the "What do we learn from this prototype? What benefits does it bring to the project's goals?" form questions, they should describe the prototype as an ensemble of RAPID-MIX technologies of their choice and explain what learning the prototype would provide. We asked participants of each group to prepare and to select a spokesperson to deliver a presentation to the whole session group explaining the scenario they had prepared. Participants engaged in the scenario-building activity for about 1 hour. After that, each of the four groups presented the design rationale for the scenario based on the theme's challenge (Figure 4.4) (40 minutes). All the presentations were video-recorded, and stored in the online video-repository for internal and shared consultation, which I subsequently analysed.

(a) (b)

Figure 4.4: a) Emmanuel Flety and Mick Grierson presenting the rationale for the *Hardware processing* scenario; b) Sergi Jòrda presenting the *Music Augmentation* scenario.

4.3.4 Results

The convergent session of the co-design workshop yielded the following data:

- Four documents which participants used to describe the theme-inspired scenarios they built during the session—Figure C.8 a, b, c, and d, in Appendix C.
- Video recordings of the session and of each group's presentation of the scenario.

In Table 4.7, I provide a summary of the scenarios built by RAPID-MIX stakeholders at the covergent design session of the co-design workshop. In the following sections I provide a detailed description of each of the scenarios. In order to capture with more detail the scenarios that the four groups of participants generated, I first transcribed each group's template describing the scenario (Figure C.8, Appendix C). The video-recordings of each group's spokesperson presenting the scenario were annotated and made available to researchers (e.g. Sebastian Mealla, Atau Tanaka, Frederic Bevilacqua). Video annotations were interpreted using informal and naturalistic content analysis to complement the template transcription. Both annotations and interpretation of the videos were compiled into a working document that was circulated among the consortium. The researchers reviewed and agreed on the interpretation, and incorporated it in deliverable D2.2 (Bevilacqua et al., 2005).

Table 4.7: Summary of the scenarios built by RAPID-MIX stakeholders at the covergent design session of the co-design workshop

Core Use Case	**RAPID-MIX technology providers**	**Target Users**	**Technologies**	**Insights**
Biosignals in Context: Biosignal repository and exploring databases	UPF PLUX ORBE	Students Researchers Developers Advertising Agencies Physicians Personal Trainers Athletes	Repovizz Teclepathy BITalino JUCE MAVEN MAXIMILIMAN IAE Essentia	Understanding how to build infrastructure with end-to-end multimodal machine learning pipelines and multimodal data repository. Extending traditional tracker services with biosensing and MIR features on mobile clients.
Processing in Hardware	GS IRCAM RS	Music hardware companies and designers musicians and hobbyists (makers)	IRCAM RIoT Maximilian Essentia	Identifying the optimal assortment and configuration of RM technologies for developing a small form factor, low energy consumption, embedded platform, capable of running RAPID-MIX DSP & ML algorithms in real time.
Music Augmentation: Designing expressive interactions	GS MTG IRCAM	RAPID-MIX researchers	The whole set of RAPID-MIX technologies and 3rd party hardware	Several independent partial prototypes to i) synthesize common infrastructure needs, ii) breakdown the problem of how to make non-performative actions become expressively interactive
Web apps	IRCAM ROLI ORBE	Developers of scalable, situated, multimodal procedural experiences that use rich input, real-time interaction and data visualization.	Physical layer: JUCE, BITalino and third party sensors; Web layer: Repovizz, CoSiMa (client); Maven, Essentia (server), BITalino, Max/MSP, Resolume	Identifying challenges in integration, cost effectiveness, market fitness and user experience for web-based designer tools.

4.3.4.1 Scenario 1 – Biosignals in Context

The text box below presents a transcription of the document (Figure C.8 a, Appendix C) which participants filled out to document their scenario.

Theme: Biosignal Repository and Exploring Databases

Group members:

Hugo Silva (PLUX)

Panos Papiotis (MTG/UPF)

Sebastian Mealla (MTG/UPF)

Andrés Bucci (Reactable)

User – Who is the user? What is the market space for the product?

Students – storing & retrieval of example data.

Athlete (Amateurs + pros) – history data term + sharing + features + statistics of self.

Researchers – same as students + batch processing.

Developers – programmatic access to data + features + SDKs + Dynamic resources.

Advertisement Agencies – same as developers + data mining + crossing bio-signal data with users profiles.

Physicians – same as Athletes and Researchers.

Composers/Artists – Same as students + features, multimodality, SDK and dynamic data.

Technology – What technologies are involved? How are they combined?

Repovizz, Teclepathy, BITalino, JUCE, MAVEN, Maximilian, IAE, Essentia

Prototype – What do we learn from this prototype? What benefits does it bring to the project's goals?

"Biosignals in Context" We add additional levels of information (i.e. biosensing, music) compared to the existing activity trackers.

We provide multimodal analysis that informs about the effect of musical content in physical performance (RUNNING).

This is a particular instance of a broader problem: Biosignals in Context.

Observations from the video footage:

The group of participants who signed up for scenario building inspired by the theme *Biosignals Repository and Exploring Databases* proposed a scenario for innovating "traditional" sports tracker services (e.g. *Runkeeper*[5], *Strava*[6]). The scenario encompassed a multiplicity of domains and use cases, and the innovation consisted of the provision of infrastructural support for richer interaction

[5] *Runkeeper*, "Runkeeper® app helps people get out the door and stick with running.", https://runkeeper.com/,

[6] *Strava*, "The #1 app for runners and cyclists", https://www.strava.com/

with mobile client applications for running and other sports. Through the addition of information layers provided by biosignal sensing, and music and audio analysis, participants proposed to support, for instance, new forms of interaction with mobile audio content and measurement of the impact of mobile audio content for advertising purposes.

Based on this scenario, participants proposed a prototype as an instance of the broader problem "Biosignals in Context". They sketched a pipeline—see Figure C.8 a), Appendix C—which described a combination of bio-signals with different kinds of information and multimodal data (e.g. ECG, EDA, accelerometer data using BITalino, audio that is listened to, and geo-location). In a processing layer, data would be synchronized and aligned, and high-level features would be extracted—e.g. heart rate variability—using Essentia. Interdependence analysis would be performed to seek for linear and non-linear correlations between the different kinds of data. Interestingly, the use of machine learning appears as implicit here (referred to as multimodal analysis, interdependence analysis for correlations). The results would be attached to the actual data as metadata or annotations and uploaded to Repovizz. A mobile client would then be developed to explore how data changes in the context of running, and how users get affected by the features of music, velocity and location, for instance.

4.3.4.2 Scenario 2 – Processing in Hardware

The text box below presents a transcription of the document (Figure C.8 b, Appendix C) which participants filled out to document their scenario.

Theme: Processing in Hardware

Group members:

Mick Grierson (Goldsmiths)

Emmanuel Flety (IRCAM)

Carles F. Julià (MTG/UPF)

User – Who is the user? What is the market space for the product?

Music hardware Company/Designer

Musicians

Hobbyists

Technology – What technologies are involved? How are they combined?

Wekinator/Gesture Follower

Gesture Agents

MAVEN/Essentia

Maximilian/JUCE

Machine learning, feature extraction for improved signatures [of audio] and interaction

Prototype – What do we learn from this prototype? What benefits does it bring to the project's goals?

We will create an embedded platform capable of running RAPID-MIX algorithms in real-time for music applications. We will create a net-audio DSP layer as a second stage. It must be small and of low-energy consumption.

Observations from the video footage:

Participants proposed a scenario inspired by the theme *Processing in Hardware* to provide insights into the optimal assortment and configuration of RAPID-MIX technologies for creating a low-latency, low-energy, small form-factor embedded platform for real-time for musical applications. This scenario targeted a user group comprising designers from music hardware companies, music hobbyists, and general users. In this scenario, the RAPID-MIX API would provide means for customising the hardware according to users' needs. Participants proposed a prototype consisting of an embedded hardware platform which was intended to support rapid prototyping, firmware uploads—including RAPID-MIX API algorithms that process and extract signatures from audio and multimodal interaction data—integration and connection with other devices for music making.

Figure C.8 b) (Appendix C) depicts the scenario in which the user customises hardware frameworks through the deployment of custom feature extractors and machine learning models. Participants envisioned a workflow where the user would use a tangible interface that hosted a board (such as a carrier boards with a socket), to which algorithms developed using the RAPID-MIX framework would be cross-compiled and flashed into. The board would provide audio, gesture and haptic signatures, and would be able to communicate them in real-time to a set of neighbouring devices. The board was also intended to stream audio or provide an audio link to these devices.

4.3.4.3 Scenario 3 – Music Augmentation

The text box below presents a transcription of the document (Figure C.8 c, Appendix C) which participants filled out to document their scenario.

Theme: Designing Expressive Interactions

Group members:

Sergi Jordà (MTG/UPF)

Frederic Bevilacqua (IRCAM)

Rebecca Fiebrink (Goldsmiths)

Francisco Bernardo (Goldsmiths)

User – Who is the user? What is the market space for the product?

RAPID-MIX partners

Technology – What technologies are involved? How are they combined?

Sensing: bio (heart, GSR) accel (eg. on feet), smell, GPS

DB [database]: environment (eg. pollution) maps/GIS data

ML [machine learning]

Feat Ext. [feature extraction] (analysis/synth)

(everything else)

Prototype – What do we learn from this prototype? What benefits does it bring to the project's goals?

provide for a thoughtful balance between:
agency/transparency
aesthetics/quality
sound/interaction
Several independent partial prototypes:
non-performative actions become sonified/expressively interactive.
- Running –> personalised soundtrack of remixed music collection
- Bus trip –> remix of ambient sounds during travel
- Others –> walk, cook, yoga...

Observations from video footage:

Participants built a scenario around the theme *Designing Expressive Interaction*, which they subsequently titled *Music Augmentation*. In this scenario, participants proposed to better understand user and infrastructural needs for different applications of interactive sound, such as sonification of non-performative activities and musical augmentation of daily routine experiences. Specifically, participants used the scenario to propose a set of independent and partial prototypes that leveraged on sensing of physiological and human movement signals, which could provide insights about how non-performative actions could be sonified and made expressively interactive. These insights were intended to inform use cases involving sonic augmentation of non-performative activities, such as running, bus trips, walking (in a gaming context), biking, cooking, and yoga. Prototypes would leverage on sensing of physiological and human movement signals, and on users' motivations and abilities, such as the ability to make music and the motivations to improve running/fitness and social cohesion—e.g. bringing the group together, giving a sense of group belonging, and promoting sharing with friends. Participants considered this scenario especially appealing to RAPID-MIX partners working in research and interested in exploring high-level problems such as agency, transparency, aesthetics, and qualities of interactions. On the other hand, participants also considered commercial applications on top of this scenario—for instance, personalised soundtracks for running, remixes of ambient sound for daily routine activities, such as commuting, cooking and yoga.

4.3.4.4 Scenario 4 – Web Applications

The text box below presents a transcription of the document (Figure C.8 d, Appendix C) which participants filled out to document their scenario.

Theme: Web Applications

Group members:
Hugo Silva (PLUX)
Panos Papiotis (MTG/UPF)
Sebastian Mealla (MTG/UPF)
Andrés Bucci (Reactable)

User – Who is the user? What is the market space for the product?
Students – storing & retrieval of example data.
Athlete (Amateurs + pros) – history data term + sharing + features + statistics of self.
Researchers – same as students + batch processing.
Developers – programmatic access to data + features + SDKs + dynamic resources.
Advertisement Agencies – same as developers + data mining + crossing bio-signal data with users profiles.
Physicians – same as Athletes and Researchers.
Composers/Artists – Same as students + features, multimodality, SDK and dynamic data.

Technology – What technologies are involved? How are they combined?
Repovizz, Teclepathy, BITalino, JUCE, MAVEN, Maximilian, IAE, Essentia

Prototype – What do we learn from this prototype? What benefits does it bring to the project's goals?
"Biosignals in Context"
We add additional levels of information (i.e. biosensing, music) compared to the existing activity trackers.
We provide multimodal analysis that informs about the effect of musical content in physical performance (RUNNING).
This is a particular instance of a broader problem: Biosignals in Context.

Observations from the video footage:
In the *Web Applications* scenario, participants advocated for the potentials of the mobile world wide web to be applied to the design of interactive and situated experiences in advertising, art and sound. Participants proposed to employ RAPID-MIX technologies to create design tools which could build on the web to provide both scalable and customisable experiences that used elements of locality and situation (e.g. using the local context acquired by phone sensors, such as GPS,

accelerometer, etc). Participants envisioned building these design tools on top of a back-end infrastructure comprised of RAPID-MIX technologies, such as Repovizz and COSIMA, and external technologies, including web standards and other web services. With the proposed prototypes of this scenario, participants aimed to obtain insights into the integration of technologies and to better understand the user experience of designer tools—specifically for making web-based designer tools fluid, efficient, creative, and fun—and about customisation and personalisation of users' experiences. Participants identified as the main challenges of this scenario latency issues, the cost of re-deployment, and the proximity between development and deployment.

4.3.5 Discussion

4.3.5.1 Deploying a first instance of a UCD action with RAPID-MIX consortium stakeholders

According to Preece et al. (2015), successful product design should consider a range of stakeholders. With the co-design workshop sessions, I intended to create a collective environment for RAPID-MIX consortium stakeholders which could stimulate their creativity, playfulness and "absorptive capacity" (Cohen and Levinthal, 1990). I wanted to induce "knowledge spillovers" (Acs et al., 1994) between RAPID-MIX stakeholders around the potentials of future RAPID-MIX technologies and their users. This UCD action aimed to provide RAPID-MIX consortium stakeholders with shared perspective into the early stage of a design process that leveraged on a user-centred approach. It also helped RAPID-MIX consortium stakeholders to align their thinking regarding the design opportunity and to create a shared vision for design goals among the group. With the co-design workshop I tried to illustrate what the intended mindset of the UCD Actions framework was, and share it across the RAPID-MIX consortium, by providing stakeholders with a common experience, a shared vision and understanding of design goals, and a shared ethos for action throughout the rest of the project.

In multi-stakeholder organisations such as the RAPID-MIX consortium, the collective knowledge is diffuse and can span many different areas. Among the RAPID-MIX consortium stakeholders, there were high levels of scientific, technical, social, cultural, business and market knowledge. Using the co-design workshop as a first instance of an UCD action, I aimed to capture the RAPID-MIX consortium stakeholders' tacit knowledge—which Polanyi (2009) considered as being contextual and difficult to transfer—and their perceptions about the potentials of the different RAPID-MIX background technologies. Engaging multiple RAPID-MIX consortium stakeholders simultaneously in a co-design session, using UCD techniques such as ideation, bootlegging, storyboards and scenario building, provided ways to externalise, share and persist these different kinds of knowledge for the common understanding of the RAPID-MIX consortium stakeholders. The action promoted the sharing of knowledge and understanding about the scope of the problem space and the dimensions of *user*, *technology*, and *context of use*. This knowledge was embodied in design artefacts—i.e.

post-its, storyboards depicting scenarios of interest from divergent brainstorming, and others inspired by core themes and background RAPID-MIX technologies—and documented through video recording for subsequent analysis and discussion.

4.3.5.2 The perceived broad scope and general purpose of RAPID-MIX technologies

In the early stage of the RAPID-MIX project, the results that the co-design session yielded—a diversified set of categories of target users, contexts of use and scenarios—exposed the assumptions and perceptions of RAPID-MIX stakeholders about the general-purpose nature and broad potential for application of the portfolio of background RAPID-MIX technologies. These results indicated that RAPID-MIX consortium stakeholders' perception of the end-user and design spaces for RAPID-MIX technologies was vast, speculative, and in some cases, extended beyond RAPID-MIX's initial conceptual boundaries (i.e. gaming, music and healthcare). This is suggested by the set of diversified categories of potential users and contexts of use that resulted from the action—for instance, *Developers*, *Education & Academia*, *Healthcare & Sports*, *Art, Music, Gaming & Entertainment*, and *Industry, Automation & Monitoring*—which are familiar to the scope of the project, but also "unfamiliar" categories, such as *Agriculture* or *Politics*.

These categories of users and contexts of use were also aligned with the data from the scenarios from both the divergent and convergent design sessions. Some of the themes that I extracted from categories of users and contexts of use—for instance, *Making vs. Using* and *Proxy users vs. End users*—suggest that there was a common understanding about the potential of RAPID-MIX technologies for application in both tools (rapid prototyping and product development) and end-user products. Other themes indicated a common understanding of the potential for application of RAPID-MIX technologies in professional and recreational contexts (*Work vs. Recreational*) and in different social contexts (*Public/Social vs. Personal/Intimate*). All of this reveals about RAPID-MIX stakeholders' perceptions of the general-purpose nature and broad potential for application of RAPID-MIX technologies.

These results were interesting, particularly from a perspective which considered the design process as an opportunity to evolve from a current state to a desired state through design. These results revealed the perception of RAPID-MIX stakeholders of the initial state of the design challenge posed by RAPID-MIX. This state included a broad and highly unconstrained design space, where a set of disparate technologies enabled a multitude of possibilities in terms of target users, contexts of use, use cases and implementations. This was of course one problematic aspect of the overall design challenge in RAPID-MIX. At the time, what this unconstrained space revealed to me was that, to better support the RAPID-MIX consortium to achieve its operational goals, my most relevant contribution could be to help finding an adequate set of constraints and guidelines for the design exploration process. On one hand, this would include helping to better understand design constraints associated to the target users, by narrowing the scope of the user space and deepening our understanding of users. On the other, it would entail the identification of useful technological design constraints and understanding their impact, as the design space of RAPID-MIX technologies and their applications evolved.

4.3.5.3 A pattern of real-time, multimodal data flows on machine learning pipelines

One overarching theme in the scenarios of both co-design workshop sessions concerned a recurrent pattern of a pipeline. The pipeline can be understood here as a model, or a metaphor, for a linear combination of stages or elements which encapsulate major system functionalities. This pattern is often used in real-time signal processing, computer science, and creative domains such as music technology and New Interfaces for Musical Expression. In some scenarios, participants did not refer to a pipeline explicitly, but rather to a sequential logic and arrangement of constituent technologies, both in abstract terms—in the divergent thinking scenarios e.g. *Le Cook Rapide*, *Intelilight* and *Guitar 911*—or using combinations of specific RAPID-MIX technologies—in the convergent design scenarios e.g. *Music Augmentation.* In some scenarios, participants drew a sketch with a pipeline—e.g. *Biosignals in Context*, *Processing in Hardware*, and *Web Applications.* These scenarios suggested a pipeline pattern to structure three main functional elements: a) different sources of real-time multimodal data flows, typically sensors; b) real-time digital signal processing and machine learning for processing the data flows; and c) mapping of the processing results to multimedia outputs. Often, but not always, participants included a data repository for multimodal data or multimedia content as yet another common element of this pipeline pattern.

In most of the scenarios, participants conveyed their interest in using combinations of different types of data and data sources. Participants provided a more or less comprehensive list of sensors and signals they found interesting to use as multimodal data sources. These sensors and signals included, for instance, biosignals—e.g. ECG, EEG, EMG—or motion, sound and video data. Other data sources with different types of data included environmental or geographic databases. Some of these data sources were depicted implicitly or integrated within a specific interaction metaphor in the scenario—e.g. the augmented cooking utensils of *Music Augmentation*, or the hand exoskeleton controller and actuator in *Guitar 911.* In many of the scenarios, participants also described high-level information, i.e. features that could be extracted from the different kinds of data and which could be useful for different use cases—e.g. heart rate variability (*Biosignals in Context*), arousal and perspiration from physiology data (*Biosignals in Context* and *Music Augmentation*), footstep rate and phase, movement technique and coordination patterns from motion data (*Music Augmentation*), or audio and gesture signatures (*Processing in Hardware*). Participants considered combinations of multiple sensors and features, with concentration around biosignals and human activity data.

Participants envisioned machine learning applied for real-time processing of the different sources of data flows, throughout the different scenarios. For instance, in the scenario *Le Cook Rapide*, participants conceptualised an interactive machine learning workflow in which end users would record the data from performing with augmented cooking utensils and use it to train models which would augment cooking recipes, for the classification of the recipe's augmented performance. Similarly, in *Guitar 911* machine learning would be employed to classify the performance and hand technique of a practising guitarist. In *Music Augmentation*, machine learning was intended for related purposes, that is, for the classification of end-user physiology and movement data, but in this case, applied to the sonification of daily routine activities and non-performative tasks. In *Intelilight* and *Biosignals in Context*, machine learning was applied to the classification and interdependency analysis of biosignals data and other multimodal data. In *Processing in Hardware*, real-time machine learning was intended for audio and gesture fingerprinting processes running in embedded hardware. These examples show a concentration of cases where machine learning is applied to real-time processes with multimodal data, many of them related to biosignals and human embodied performance interaction.

Participants envisioned applications of machine learning tasks to real-time multimodal data flows, mainly for building direct mappings to multimedia outputs, and for classification of context variables and human activities in different scenarios. The *Music Augmentation* scenario depicted a use case in which different kinds of multimodal data, features, and machine learning could be applied to building mappings for sonification of disparate daily routine and non-performative activities. Similarly, in the scenario *Le Cook Rapide*, the multimodal data sources consisted of augmented cooking utensils, i.e. augmented with sensors for users to manipulate and interact with the system in real time. In this scenario, machine learning was intended for building mappings between the end users' embodied interaction and multimedia outputs (audio and visuals) and to evaluate the performance cooking a specific recipe.

Along the same lines, in the *Guitar 911* scenario, real-time data flows originating from multiple sensors integrated in a hand exo-skeleton would be classified using real-time machine learning for the evaluation of technique in musical performance, and drive the specific tuition needs through immersive media content and haptic feedback. In *Intelilight* and *Biosignals in Context*, participants depicted the integration of real-time multimodal data flows from multiple sensors and data sources, with a cloud-based multimodal data repository service. In the former scenario, biosignals classification would be applied for real-time analysis of context variables related to patients' health status. In the latter scenario, the pipeline would be used to assess the impact of advertising and audio content in the physical activity of mobile client application end users. In both scenarios these pipeline segments would provide for information requirements of multimedia client applications.

In general, the main areas of interest for future applications, as conceived by participants, appeared to fall within the classification of context variables and human activities, and building mappings for expressive parameters of real-time audiovisual content. To achieve this, participants envisioned real-time, end-to-end, multimodal machine learning pipelines that were mostly agnostic to data and sensor configurations. In the different scenarios, the constituents of the conceptual pipeline appeared to have some aspects better defined than others. For instance, the application of machine learning as real-time process appeared to be a common and well-defined factor according to the visions that participants expressed in the scenarios.

Other general aspects of machine learning, however, which concerned system architecture and workflows, for instance, were more unclear and ambiguous. In terms of system architecture, in some scenarios, participants conceived the machine learning element as local, interactive, and applied to processing of real-time multimodal data flows. In other scenarios, machine learning was an element of a server-side process applied in a multimodal data repository service that collected real-time data and issued real-time notifications for client applications to consume. In terms of workflow there was one case, *Le Cook Rapide*, in which the workflow was made explicit and devised as interactive machine learning. In most of the scenarios, however, participants focused on the machine learning functionality rather than on the workflow. The machine learning workflow was made implicit and potentially assigned to experts or system developers.

4.3.5.4 Better understanding common infrastructural software needs in RAPID-MIX

One overarching theme in the scenarios of the convergent design session concerned RAPID-MIX consortium stakeholders' need to better understand the common infrastructural software needs of future RAPID-MIX technologies. Infrastructural software, also known as middleware, can be generally understood as software that supports common development and operation requirements of other software, such as end-user applications, or other middleware for end-user applications (Edwards et al., 2003). Infrastructural software typically includes, for instance, software libraries, online services, toolkits for software development, or other platforms.

At the time of the co-design workshop, there were several technologies in the RAPID-MIX portfolio that qualified as infrastructural software—e.g. software libraries such as XMM, GVF, CoSiMa, IAE and Maximilian, or cloud-based repositories and web services, such as Repovizz and Freesound.org. Also included in this category was JUCE, a software development framework. The RAPID-MIX API qualified, most notably, as the unifying piece of infrastructural software that was intended to fulfil the needs of future RAPID-MIX technologies. The RAPID-MIX API was also part of future RAPID-MIX technologies, alongside the MIX products, and any other applications and prototypes that would be built as clients of the RAPID-MIX API as infrastructural software.

Throughout the scenarios of the convergent design session, participants proposed several prototypes that integrated RAPID-MIX technologies. This included prototypes for understanding RAPID-MIX technologies both individually, as core components, as well as integrated and structured by the infrastructural software. For instance, in *Biosignals in Context* scenario, participants proposed prototypes for understanding how to build an infrastructure that integrated specific RAPID-MIX technologies—i.e. with BITalino, GVF, Repovizz—in end-to-end, multimodal machine learning pipelines with a cloud-based data repository. These prototypes were not meant only for specific use cases and applications of their interest, but also as a means to obtain insights for these infrastructural software needs.

For instance, in the scenario *Processing in Hardware*, participants proposed prototypes for a use case around their vision of infrastructural software serving embedded hardware applications. Infrastructural software would provide DSP and ML code to be flashed as firmware to prototypes built of embedded hardware. This would entail building infrastructural software with features for operating under the constraints of that specific environment (e.g. low-level, portable software). Similarly, in use cases related to the *Music Augmentation* and *Biosignal in Context* scenarios, an infrastructure would have to fulfil mobile client applications' requirements and potentially server-side application requirements, in terms of specific feature extraction and machine learning algorithms to. In the *Intelilight* and *Web Applications* scenario, infrastructural software would need to provide for the requirements of server-side applications and installations.

The diversity of prototypes and use cases that participants proposed across scenarios suggested a considerable diversity of infrastructural software design features. The infrastructural software would need to include support for different implementations, run-time environments, languages, performance requirements, etc. These suggested immediate requirements or design features that could be extracted for infrastructural software. For instance, infrastructural software would need to be built with cross-platform and multi-target capabilities. Additionally, based on the insights from Section 4.3.5.3, the infrastructural software would need to provide ways to structure multiple instances of pipelines, with different combinations of end points and components.

Understanding the common infrastructural software needs posed one of the central conundrums for RAPID-MIX consortium stakeholders. Based on how the participants expressed their intention of obtaining insights about common infrastructural needs in the scenarios, I extracted the following three main points:

- Understanding what the constituent elements of infrastructural software should be and how they should be structured—for instance, in the *Biosignals in Context* scenario, participants conveyed the need to understand how to build an infrastructure which integrated real-time, end-to-end, multimodal machine learning pipelines with a cloud-based data repository, with RAPID-MIX technologies. In the *Processing in Hardware* scenario, participants highlighted the need to identify the optimal assortment and configuration of RAPID-MIX technologies to build different real-time, sensor-based embedded hardware applications.
- Understanding which features of infrastructural software were required to support the different use cases and applications envisioned by RAPID-MIX consortium stakeholders—in the *Music Augmentation* scenario, participants communicated the need to understand and synthesise common infrastructure needs for different applications for sonification and musical augmentation of daily routine and non-performative activities.
- Understanding how the features of infrastructural software should be designed without knowing about future client applications' requirements—this was necessary because of requirements of future client applications from infrastructural software end users, both from inside

> and outside the RAPID-MIX consortium. In the *Web Applications* scenario, for instance participants mentioned the need to identify and understand aspects of the infrastructure that were necessary to support different web-based designer tools, such as integration challenges, end-user experience (i.e. of developers' experience as users of the infrastructural software), and the cost effectiveness provided by the re-use of the infrastructural software.

The challenges of infrastructural software design are usually associated with the different requirements which client applications of infrastructural software may have. This problem may be aggravated in situations, such as in RAPID-MIX, where there were multiple stakeholders with disparate use cases for client applications of the same infrastructural software, and the design of the infrastructural software was constrained by an initial set of disparate sourcing technologies. The findings from this UCD action were effective for uncovering these problems and focusing the directions of subsequent research.

4.4 Gathering user needs in Sónar+D 2015 Music Hack Day

I collaborated in a UCD action deployed at the Sónar+D 2015 Music Hack Day (Barcelona, June 18-19th, 2015), which RAPID-MIX consortium partners MTG/UPF led. Sónar+D is a space for creativity and innovation within the electronic music festival Sónar, which includes "talks, demos and workshops, tech shows and exhibitions, immersive experiences, [one-on-one] mentoring and community meetups, and much more"[7]. Music Hack Day[8] (MHD) is a series of globally dispersed 24hr-hackathons focused on music technology which started in London, in July 2009. Briscoe and Mulligan (2014) defined a hackathon as "an event in which computer programmers and others involved in software development collaborate intensively over a short period of time on software projects" (p. 1). Briscoe and Mulligan also described hackathons as a "contest to pitch" (p. 1) and "present instances of prototype digital innovation" (p. 1), and as having the "the provision of an award or prize which adds a competitive element [...] (often sponsorship for further development)" (p. 2).

Music Hack Day was chosen for its alignment with RAPID-MIX in terms of audience, domain, and type of event. The target audience included developers, designers, engineers, researchers, musicians, creative coders, makers, among other music technology enthusiasts and professionals in the creative industries. The format of participation in MHD was also convenient to a UCD action given that it entails rapid prototyping music technology, including applications, tools or new interfaces, with the set of technical resources (APIs, prototyping platforms, etc.) that the event's technology sponsors provides.

My goals for the UCD action at Sónar+D 2015 MHD overlapped with the RAPID-MIX consortium's goals. Namely, these goals were:

[7] Sónar+D, https://sonarplusd.com/en/about

[8] Music Hack Day, https://en.wikipedia.org/wiki/Music_Hack_Day

- To introduce RAPID-MIX technologies to a group of participants which matched the profile of lead users (von Hippel, 1986) and early adopters (Rogers, 2003) (see Section 2.2.1) of RAPID-MIX API and future RAPID-MIX technologies.
- To promote the use of RAPID-MIX technologies by participants in the process of rapid prototyping in the 24-hour hackathon.
- To gather insights on participants' experience with early RAPID-MIX technologies in rapid prototyping at the hackathon, and on the prototypes that would result from that process.

RAPID-MIX consortium stakeholders from MTG/UPF organised MHD as a 24-hour hackathon within Sónar+D 2015, with parallel activity tracks for different projects and technology providers. They organised a special activity track on the topic of "Wearables and Music Performance" to promote the use of RAPID-MIX technologies and set up a challenge with a prize (undisclosed at the time) to motivate participation. This activity track was announced on the event's official website as an activity for hands-on creativity and prototyping around music creation and performance using biosignals and human motion sensing, interaction design and wearable interfaces. MTG/UPF published this announcement on the event's official website[9] about two months before the official date to attract participants' applications in a timely fashion. Furthermore, MTG/UPF organised a series of pre-event workshops and hands-on sessions on background RAPID-MIX technologies with other RAPID-MIX stakeholders.

I collaborated in the deployment of this UCD action with MTG/UPF and other RAPID-MIX stakeholders as technology providers of the RAPID-MIX challenge at Sónar+D MHD. Specifically, my contributions to the UCD action included collaborating in:

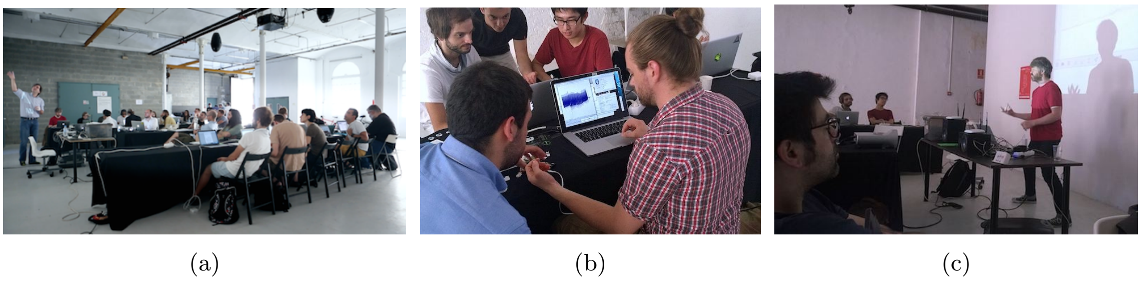
- the design of the interviews, preliminary offline surveys, and a follow-up online survey (see Appendix D) for the hackathon.
- the operational tasks, which included engaging with participants and selecting candidates for interview, based on the integration of RAPID-MIX technologies on the prototype they produced at the hackathon, and subsequently administering interviews.
- the subsequent organisation and analysis of data.

In the following sections I detail the procedure, the results and the main insights that we gathered from this UCD action.

4.4.1 Pre-event workshop procedure

The pre-event activities involved thirty participants in induction and hands-on sessions with early RAPID-MIX technologies at Hangar (Barcelona, June 17th, 2015). Tables D.1 and D.3 (Appendix

[9] http://musichackday.upf.edu/mhd/2015/



(a) (b) (c)

Figure 4.5: The MHD pre-event workshop at Hangar a) Hugo Silva presenting an introduction to physiological computing and biosensing, b) participants in hands-on exploration with BITalino and OpenSignals, and c) Mick Grierson demonstrating Maximilian.

D) present the induction and hands-on sessions scheduling, syllabuses and the RAPID-MIX stakeholder who led each session. This workshop provided a gradual introduction on how to create a pipeline using a set of RAPID-MIX technologies—i.e. BITalino, Wekinator, GVF, Maximillian and JUCE—to rapid prototype wearable mobile music interfaces and instruments using biosignals and motion sensing. The workshop sessions included:

- Introduction to Physiological Computing and Biosensing (30 minutes)—Hugo Silva (PLUX) presented an overview of the different biosignal data sources that BITalino can provide, focusing on muscle, heart and sympathetic nervous system data, and how to capture data with BITalino and use it in applications (Figure 4.5a).
- BITalino hands-on (90 minutes)—participants self-organised in groups to explore BITalino and Opensignals software (Figure 4.5b).
- Introduction to interactive machine learning (30 minutes)—Rebecca Fiebrink (Goldsmiths) presented an overview of interactive machine learning, supervised learning algorithms, and its application to real-time interactive systems.
- Wekinator and GVF hands-on (60 minutes)—participants explored how to apply interactive machine learning with BITalino data with Wekinator and GVF to to create gestural controllers and musical instruments with examples of common music environments (e.g. Max/MSP, PD, ChucK, SuperCollider) from the Wekinator toolkit.
- Introduction to sonic interaction design for mobile and wearables (30 minutes)—Mick Grierson (Goldsmiths) presented an overview of Maximilian feature extraction and sound design libraries for applications for mobile sound processing and interaction (Figure 4.5b).
- Maximilian and JUCE hands-on (60 minutes)—participants explored how to integrate Maximilian with the pipeline assembled in the previous sessions, using Wekinator/GVF features computed from BITalino data, for creating sonic interactions.

(a) (b) (c)

Figure 4.6: RAPID-MIX at Sónar+D 2015 Music Hack Day: a) and b) participants working in the 24-hour Sónar+D MHD hackathon, and c) the early stage of one of the 'hacks'.

4.4.2 Sónar+D 2015 Music Hack Day procedure

The following day, I joined other RAPID-MIX stakeholders to participate in Sónar+D 2015 MHD (Barcelona, June 18th, 2015) as technology providers in the hackathon. One hundred people participated in the hackathon and self-organised in groups. Before the event started, there was a general debriefing and presentations by the technology providers about available technologies. The debriefing covered general aspects of the hackathon such as reminders about the different thematic tracks, rules, and logistics of the event (e.g. free pizza and drinks, sleep over zones). Participants were requested to prototype a digital artefact (e.g. an app, interface, web page, physical artefact with embedded hardware) in 24 hours, after which they would have to present it to a jury comprising the tech providers for evaluation.

After the briefing, RAPID-MIX stakeholders joined other technology providers in a set of short presentations about the technologies that were available for participants to use. RAPID-MIX stakeholders presented the same set of RAPID-MIX technologies that as used in the workshops held the previous day—i.e. BITalino, Wekinator, GVF, Maximilian and JUCE—for the "Wearables and Music Performance" thematic track. For this thematic track, participants of the Sónar+D Music Hack Day Hackathon were challenged to develop prototypes that integrated RAPID-MIX technologies, using biosignals, motion sensing, machine learning for music creation and performance. Participants were free to choose technologies from any technology provider.

Once the event started, the RAPID-MIX consortium stakeholders were available for the provision of assistance and technical support. As participants engaged in prototyping and requested assistance, RAPID-MIX consortium stakeholders would attend to different issues—mostly installation and configuration issues, such as the BITalino's physical setup. As the groups of participants worked on their hacks, we carried out a preliminary survey (see Section D.2, Appendix D) across all groups to screen the hacks for RAPID-MIX technologies integration. In this preliminary survey, we identified the members of the group, their chosen spokesperson and specific requirements for the presentation, and the constituent technologies of the prototype. From the 42 groups of participants actively engaged in the hackathon, we identified 10 groups which had integrated RAPID-MIX technologies into their prototypes. We selected these groups and scheduled interviews with the groups' spokesperson, some for later that day, and the remainder for the next day before the final

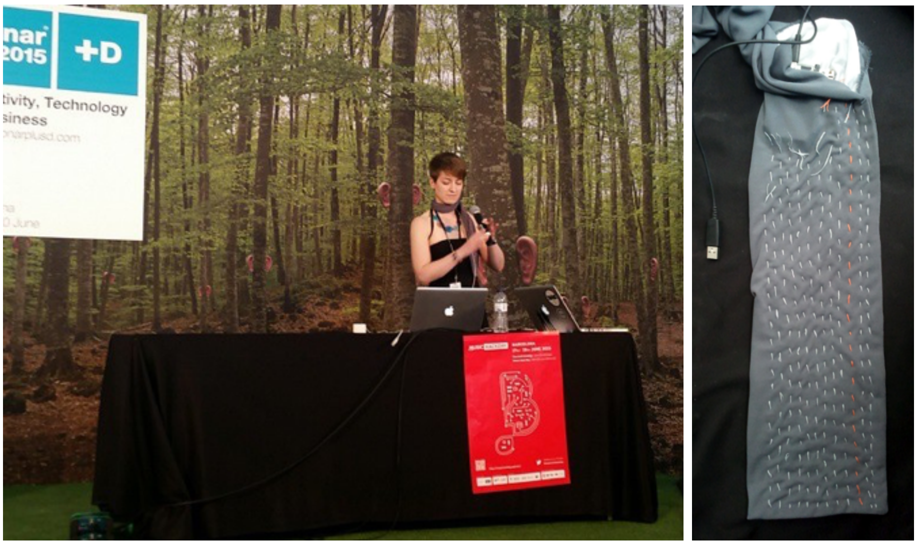


Figure 4.7: a) The winner of the RAPID-MIX design challenge at Sónar+D 2015 Music Hack Day 24-hour hackathon presenting b) the winning ‘hack’, a new musical interface that comprised a scarf with conductive thread embroidery and a microcontroller, employing the IML workflow with Wekinator for mapping gestures to audio sample playback.

presentations. Rebecca Fiebrink, Mick Grierson and I did parallel interviews with participants (20 to 25 minutes each interview). We held these interviews with each group’s spokesperson (14 participants, 13 males and one female), and recorded audio and video. At the end of the interview, we asked participants to fill out a follow-up online survey (see Section D.3).

As the 24-hour time limit became due, the MHD organisation prompted the teams to prepare for presentations of the hacks and started the session. Participants presented their hacks on the stage to the audience and the other groups (approximately 140 minutes). All presentations were video recorded, streamed online and published on YouTube[10]. After the presentations session, the jury of each thematic track convened to select the winners of that track. The winner of the RAPID-MIX challenge, *Project Jane* (Figure 4.7), received 3 BITalino kits.

After the UCD action, I was in charge of data management and analysis. I stored the entire video interview collection in the video repository to provide shared access to the RAPID-MIX consortium. I used a naturalistic approach for annotation of the videos according to interview questions and participants answers. I also carried out a rudimentary and manual content analysis to extract themes. The annotations and themes were validated by Goldsmiths’ researchers and the authors of the deliverable *D2.2 Design Guidelines for Prototyping* (Bevilacqua et al., 2015).

4.4.3 Results

The online Sónar+D 2015 follow-up survey (Appendix D, Section D.3) asked participants about a) demographics, b) their qualifications, experience and skills, c) feedback about RAPID-MIX

[10] 2015 Music Hack Day: Presentation of Hacks https://youtu.be/sWerNCeb7JE

technologies, and d) their perspective concerning online communities. Thirteen participants (twelve males, 1 female, mean age 27) responded to the surveys. Figure 4.8 below shows basic statistics from the data gathered covering professional activity, programming experience, preferred languages and use of RAPID-MIX technologies.

Figure 4.8a shows the sample distribution according to their regular activity. Most of the hackers using RAPID-MIX technologies were students (61.5%) including undergraduate, master students and PhD students. As shown in Figure 4.8b, the majority had advanced and expert programming skills. Figure 4.8c shows that the main preferred programming languages were Max/MSP (25%), C++ (20%) and PureData (15%).

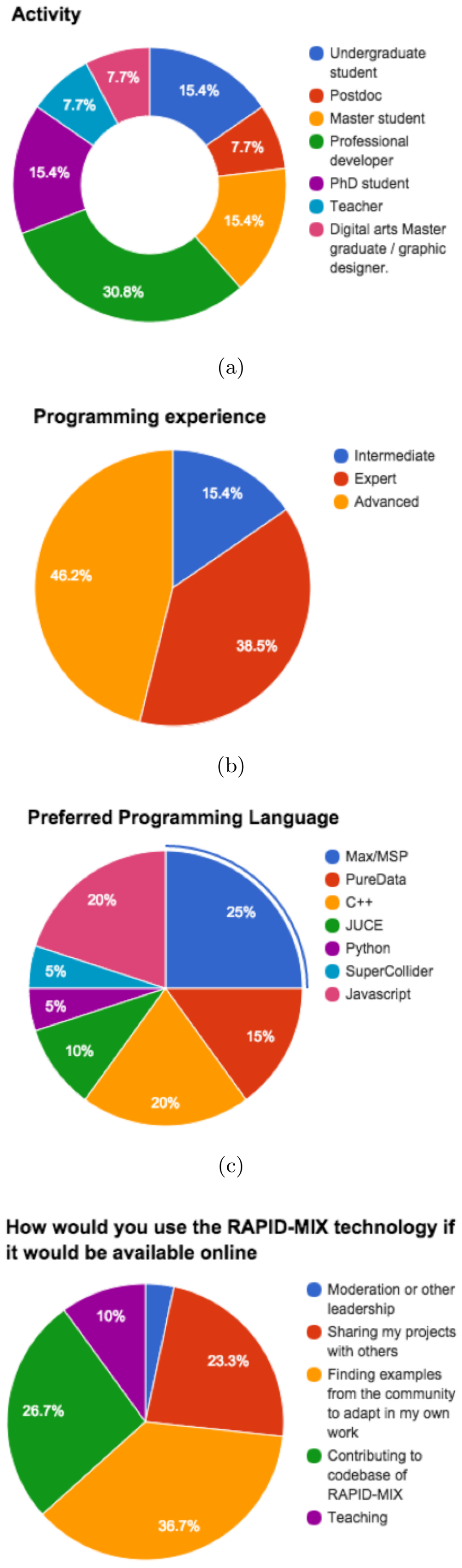
When asked about the willingness to use RAPID-MIX technology if made available in an online community, 61.5% of participants declared that they would be likely to use it. Figure 4.8d shows types of participants' expectations of uses for future online RAPID-MIX technologies. Activities related to the community aspects of RAPID-MIX stood out as favourites, such as finding and adapting code examples available on the platform (36.7%), contributing to the codebase (26.7%) and sharing projects with others (23.3%).

We carried out semi-structured interviews with participants, asking which technologies they tried in the hackathon, how they described their prototypes and which technologies they used in their prototypes. We also asked about their motivation for building their hacks and the main impediments encountered. We also asked participants to speculate on possible uses for the RAPID-MIX API and future MIX products. End users provided the RAPID-MIX consortium with invaluable insights concerning the following aspects:

a) *Knowledge about the target users' background and technical skills* – students predominated among the hackers, ranging from undergraduate, to Masters, through to PhD students. There were also developers who worked professionally with audio, sensors and wearables. Overall they presented a variable range of programming proficiency, both in rapid prototyping environments (e.g. PureData, Max, and SuperCollider) and more production-oriented languages (e.g. C++, Python, JavaScript).

b) *Perceived advantages and limitations of RAPID-MIX technologies* – in general, there was very positive feedback about the technologies. Many participants stated they provided good support for expressivity, robustness, simplicity and user-friendliness. Notably, the machine learning tools (Wekinator and GVF) were considered very helpful for exploration and rapid prototyping of expressive interfaces based on real-time biosignals and motion data (P1, P3, P4). There were also specific limitations indicated in the BITalino hardware (i.e. noise, difficulties in obtaining data, in setting up a specific configuration of hardware with accelerometers)(P4, P7). One participant (P7) referred to the power of the RAPID-MIX API for removing skill barriers in working with sensors.

c) *Categories of prototypes which indicate potential areas and products of interest* – there was



(a)

(b)

(c)

(d)

Figure 4.8: Basic statistics for hackathon follow-up surveys (Bevilacqua et al., 2015)

a significant concentration of prototypes in the category of *Music Controllers* (Appendix D, Table 4.9). Other categories were *Game*, *Mobile App*, *Wearable*, *Recommendation System*, *Quantified-Self* and *Robot Controller.* Figure 4.7 illustrates one of the music controllers that won the RAPID-MIX challenge. Other music controllers consisted of wearable devices which used BITalino's physiological and accelerometer signals. There was also an Android mobile app for assisting in music creation based on audio input analysis, and a robotic control app based on motion detection. Participants also integrated RAPID-MIX technologies with other external technologies, including Arduino and Bare Conductive microcontrollers, Sparkfun sensors, Kinect, and also with music and audiovisual production environments, such as Ableton Live and Resolume. One participant (P3) referred to the potential of RAPID-MIX in other areas, such as IoT and home automation .

d) *Support for cross-platform development and multi-target deployment* – participants expected to be able to use RAPID-MIX technology in cross-platform development, for creating prototypes that would work quickly and seamlessly across different operating systems and devices. Some participants considered JUCE (a cross-platform development environment with multi-target deployment) as the ideal unifying framework in which to integrate the RAPID-MIX technologies such as API (P2, P6, P9).

e) *Support for open-ended exploration and appropriability* - Most significantly, participants called for features which would help lower the technical barriers and foster creativity. Feature recommended included proper documentation (P3, P4), templates, tutorials, examples (P1, P3, P5, P6, P9) and essential building blocks (P1, P5). Some participants considered it very useful to have building blocks that wrapped sensors or learned gestures (P7, P8). In general, participants considered that the RAPID-MIX API should provide these features to support open-ended exploration and to successfully provide for easy adoption.

f) *Perceived importance of a community* – there was almost unanimity regarding the importance of an online community that grows alongside the development of the RAPID-MIX API. Participants expressed the will to engage with such a community in different ways, mainly as a means to obtain examples to explore and extend (P1, P8, P10), and to place technical questions or find answers (P10). Other participants admitted using such a community to share code and to contribute to the code base (P1, P4, P6), and to help others by mentoring and answering technical questions (P4).

4.4.4 Discussion

The UCD action for gathering needs and feedback from potential RAPID-MIX end users at the Sonar+D MHD provided the consortium with interesting insights. Most importantly, these insights concerned:

Table 4.9: Hacks produced in the 24-hour hackathon at Sónar+D 2015 Music Hack Day

Hack description	**Type**	**Technologies**	**ML**
Scarf with conductive thread embroidery that supports IML workflow with for mapping gestures, poses and touches to audio sample playback	Music controller	Conductive Thread Wekinator Bare Conductive Max/MSP	Yes
Musical pendulum built with motion sensors	Music controller	Max/MSP Maximilian RIoT	No
Tonalizer to help musicians jam with non-digital instruments by recommending samples in the right key	Recommendation system	PureData Essentia JUCE	No
Theremin-like interface with conductive ink and motion sensing to support musicking	Music controller	BITalino Python Bare Conductive Conductive Paint	No
A wearable, wireless controller that uses the BITalino accelerometers, muscle activity (EMG) and conductive material for real-time audiovisual manipulation	Music Controller	BITalino Max/MSP Resolume	No
A sequencer-integrated controller for low-pass filtering using BITalino EMG signal	Music Controller	BITalino Essentia Max/MSP	No
Robotic control through gestures, using Wekinator and GVF	Robot Controller	PureData GVF Wekinator Ableton Live Essentia	Yes
Sonic interactions for workout based on multimodal data acquired with the BITalino	Quantified self	BITalino, Max/MSP Arduino Wiimote Ableton Live.	No
Interactive spinning balls incorporating LEDs and gesture recognition	Game	Arduino OpenFrameworks Freesound	No
Mobile app to transform the user's hum into a tool for performing melodies	Mobile App	JUCE Freesound Essentia	No
BITalino used for controlling pitch and amplitude of an oscillator	Music Controller	BITalino JUCE	No

a) *Lack of adoption of ML tools* – Despite having attended a workshop on using RAPID-MIX ML tools, only two groups of participants used such tools in practice or achieved successful integration in their prototypes (Table 4.9). Two participants reported that RAPID-MIX ML tools were useful for the exploration of real-time sensors. This indicated potential issues with the ML tools, which required further research to uncover the factors and limitations of adoption for the current and future RAPID-MIX ML technologies.

b) *Creative and technically-skilled users with interest in the creative industries around audio technology and multimodal interaction* – Our UCD actions attracted a majority of academics (mostly students), hobbyists and independent professional developers in creative industries, who wanted to explore audio and sensors (Table 4.8). Mainly, our potential end-users had advanced skills in both rapid prototyping and production-oriented programming languages. The range of varying skills we observed suggested that the RAPID-MIX API should provide for user diversity and support different affordances for a considerable range of users—between novice and expert developers and from hackers and makers, through to SMEs developers and designers.

c) *Ambiguity in relation to the perceived complexity and simplicity of the tools* – There was positive feedback about RAPID-MIX technology, in general, although both advantages and limitations were identified. Concerning the perceived advantages of RAPID-MIX technologies, users highlighted their simplicity and user-friendliness, the support for exploration and rapid prototyping, and their OSS license. Regarding the perceived limitations, some of the RAPID-MIX technologies were found fiddly, technically complex, and overly focused on the audio and music domain.

d) *DIY controllerism as an area which RAPID-MIX technologies could primarily contribute to* – Our UCD actions revealed a significant concentration of interest in applications of music and robotic controllers, games, mobile apps, wearables, recommendation systems and quantified-self systems (Table 4.9). On one hand, this highlighted a potential interest in DIY controllerism as an area where customisation and personalisation could benefit from new tools and for which the RAPID-MIX technologies could contribute to. On the other hand, it highlighted opportunities and potential products to be developed by the RAPID-MIX consortium.

e) *Needs for cross-platform and multi-target development and deployment toolkits* – We found a high level of heterogeneous needs in development and deployment environments. Potential end users expected to be able to use RAPID-MIX technology in different development environments and for deployment, which suggested that RAPID-MIX API should support cross-platform development and multi-target deployment.

f) *Needs for open-ended exploration and easy-to-adopt toolkits* - In general, participants requested features in RAPID-MIX API that would help to lower the entry barriers and ease adoption of working with sensors, and foster creativity through support of open-ended exploration. They suggested the provision of elements such as rich documentation and essential building blocks.

g) *The importance of a community* – Participants expressed a need and desire to engage with a technology-related user community that could accelerate their exploration, troubleshooting, and to which they could contribute with their innovations and technical knowledge.

The findings of this UCD action were varied, from the reassurance of which areas the RAPID-MIX API could best contribute to, to laying the path for the RAPID-MIX API design process, through to uncovering how the lack of time for learning and of a good user experience in the exploration of tools could impact on their perceived utility.

Participants' projects often resorted to novel applications with sensors, which required sensor integration through different combinations and different ways to make sense of sensor-based data. Thus, they welcomed many of the characteristics of RAPID-MIX prototypes. For instance, the application and design of the RAPID-MIX technologies for rapid prototyping purposes, and their availability as OSS tools.

Participants revealed enthusiasm and overall interest in incorporating RAPID-MIX technologies into their projects. Participants' perceptions were aligned with the RAPID-MIX consortium internal expectations, which contributed to reinforcing the scope of RAPID-MIX technology users. We were able to obtain an improved understanding of hackathon participants as an important RAPID-MIX target user group.

4.5 Limitations, threats to validity, and mitigation of potential bias

When considering the limitations, validity and reliability of the results in the first action research cycle, one must recognise first and foremost that both UCD actions were grounded and highly constrained within the practical context of RAPID-MIX. High-level constraints at stake included the adoption of an action research methodology as a premise at the entry to the research-client system, the requirement to deliver and deploy a working UCD methodology for participants to use in the project, and lack of full control in determining these actions.

As explained in Section 4.2.5, I did not take the initiative for the first two UCD actions. Rather, they were jointly defined by the consortium with regards to the timing of occurrence at the project stage, target participants, and location. One consequence of this constraint was that the samples of participants for both UCD actions were limited and biased. I did have the flexibility to organise the UCD actions, choose the technique to deploy which I considered fit for the action and defend it before the research group, and collaborate on the UCD actions deployment.

My evaluation of whether actions have produced intended consequences is positive, but with significant caveats. The UCD actions fulfilled the research-client goals defined for the first and preliminary action research cycle (Section 4.2.5). Other issues revealed more problematic aspects and showed that the priorities of the action researcher and research client are not always aligned. Several assumptions which I made both before and during the actions' planning subsequently failed. These assumptions resulted from instructions from the hierarchy that I organise and expose the data and analysis steps as much as possible for others to be able to have a quick 'cut through' the data and make sense of it. I had assumed that, that after the deployment of techniques and data collection, there would be a subsequent stage of collaborative data analysis and validation.

In reality, the assumptions about collaboration in data analysis, distillation and validation fell apart. I had to undertake most of the sense-making activities on my own, and resort to exposing my conclusions in reports and deliverables to obtain some kind of validation. That means that I have to account for personal researcher bias that may result from the overload of information from data, trying to find meaning from data and within a reasonable amount of time, all on my own. For instance, when clustering post-its and building the affinity map, bias such as confabulation or the clustering illusion should be accounted for. Or, accounting the confirmation bias when finding details which confirmed my beliefs, such as in finding the emergence of a pipeline pattern.

I assessed the consortium stakeholders' thinking, characterised the initial set of target users in more depth, defined contexts of use and gathered feedback and insights about RAPID-MIX background technologies. This action supported the conjecturing of scenarios, the creation of a setting for learning and understanding the goals of stakeholders and behaviour of participants. However, it is difficult to make a strong claim that one UCD action aligned participants thoughts about future opportunities for design and development. In the co-design workshop, participants of the co-design session were project stakeholders and members of the RAPID-MIX consortium. This meant that it was likely that they had similar interests in familiar domains, given their participation in a very narrow sector of technology.

Although there was structuring and scaffolding of the ideation process with storyboards and scenarios as interesting outcomes, there was no formal validation to whether that UCD action did support knowledge spillovers or the alignment of participants thinking. Some steps were taken to mitigate bias, such as ensuring the institutional diversity of the participants' groups, introducing randomisation steps to the divergent design process (e.g. randomising groups of post-its and participant-selected aggregates). Validation could be have been performed, for instance, through a survey to consortium stakeholders on the impact of the co-design workshop. However, that did not happen, mostly due to work overload and my lack of experience and that of other researchers in the group at that preliminary stage of research.

In the second UCD action, there were also intrinsic difficulties and issues related to deployment. This UCD action, which targeted the large scale MHD hackathon had a large number of participants, was organised by multiple partners and deployed immediately after the first UCD action (i.e. the co-design workshop). This sample was also biased from the recruitment process, which established desired characteristics to the participants' profile. Steps taken to mitigate bias during the UCD action included elaborating the interview scripts collaboratively, having different research team elements performing and recording the interviews to avoid individual researcher bias. This last step introduced other limitations with regards to accuracy, precision and reliability of the data collected. The contextual constraints of the hackathon (e.g. time constraints, stress, lack of sleep and energy associated with running a large scale 24hr-hackathon after full days with co-design workshops and induction sessions) may have indulged a more 'quick and dirty' approach with 'sufficient' rigour and the hope of gathering as many findings, and extract as much value as possible. As a result, one researcher did not ensure the same quality of data collection.

Steps to mitigate limitations were also taken after the action. Specifically, design artefacts and recordings were made available in an online, centralised repository. I annotated the videos and they were reviewed by different members of the research team. Design materials, analysis and distillation outcomes were reported to the consortium. However, I was not able to garner participation in the data analysis process nor explicit formal validation from the members of the consortium. Rather, this happened indirectly through the writing and internal dissemination of the deliverables. This leads to the conclusion that further organisational arrangements would be required at the stage of the research-client agreement, or of the definition of the UCD Actions framework to ensure that the validation step would be enforced.

Although it was impossible to make claims that generalise beyond the narrow confines of such a context, there was relatively solid evidence in the results with prototypes which show a lack of adoption of ML. Arguably, there was also evidence of the need to think about the common infrastructural needs for all the partners, and how it converged towards a multimodal ML pattern.

4.6 Summary of First Action Research cycle insights

In this section, I summarise the main insights of the early UCD actions in which I engaged to inform practice in RAPID-MIX and my own action research. I reflect on the success of these early User-Centred Design actions in achieving operational goals and influencing the project course of action. I reflect on the most significant outcomes to the design artefact and my research questions.

In the first action research cycle of RAPID-MIX, I was involved in the planning, deployment and evaluation of several UCD actions. The UCD actions here reported included a co-design workshop with RAPID-MIX consortium stakeholders, a large-scale hackathon at Sónar+D 2015. As firstly identified in the diagnosis stage (Section 4.1), these UCD actions pursued the goal of learning about the target users groups and the space of compelling applications for RAPID-MIX technologies, and informing the RAPID-MIX consortium with the resulting insights. More specifically, these insights aimed to inform the specification of prototypes and of guidelines for the RAPID-MIX API.

These interventions targeted different stakeholders, including internal RAPID-MIX consortium stakeholders, as well as hackers, makers, students and hobbyist developers.

From RAPID-MIX consortium stakeholders, I learned that they considered the initial portfolio of RAPID-MIX technologies as broad-scoped and general-purpose. I also learned about RAPID-MIX consortium stakeholders' early perception of the design process, departing from a broad and unconstrained space, both in terms of future RAPID-MIX technologies and their users. I found a pattern that RAPID-MIX consortium stakeholders used for structuring early visions of future RAPID-MIX technologies. This pattern, which does not differ from existing DSP and NIME mapping models, was a pipeline which structured multimodal data sources, real-time data flows, machine learning, and multimedia outputs. RAPID-MIX consortium stakeholders referred to this pipeline more or less implicitly, either in an abstract way, as concrete combinations of background RAPID-MIX technologies, or as common infrastructural software.

Additionally, I found that RAPID-MIX stakeholders' needs for common infrastructural software required further and more in-depth understanding concerning:

i) what the alternatives for creating infrastructural software would be and where would they overlap,

ii) what methods to make infrastructural software alternatives complementary, or ideally, unified,

iii) the trade-offs, the features of highest value, the pros and cons of the infrastructural software and how different directions would compete in terms of development effort and resources,

iv) infrastructural software needs of users external to the RAPID-MIX consortium.

Another goal for this stage of research was improving the identification, delimitation and understanding of the primary user groups of the RAPID-MIX API. The early UCD actions helped to refine our initial understanding of user categories (Table 4.1) into the following user profiles:

- Developers of web, mobile, games, audio software applications;
- Designers, creative coders, hackers, makers;
- Academic researchers, teachers, students;
- Artists (e.g. music performance and composition, audiovisuals);
- End consumers of products that use the RAPID-MIX API

We also obtained a deeper understanding of some of the fundamental characteristics associated of these user profiles, such as:

- background—educational and professional backgrounds in areas where computational creativity is applied to the design and production of artefacts for personal use, and for cultural and commercial exploitation (e.g. from interaction design, to sound and multimedia design, to audio and software engineering).
- technical skills—specifically, having little or no experience in ML, and different levels of programming skills in production-oriented languages (e.g. C++, JavaScript, Python) and rapid prototyping languages and environments (e.g. PureData, Max, and SuperCollider).
- development needs (e.g. ease of adoption, support of open-ended exploration, and community around the technology).
- perceived advantages (e.g. open source, online availability) and limitations of RAPID-MIX technologies (fragmentation, complexity of machine learning)

Early UCD actions also helped to reveal the space of compelling applications for stakeholders and end-users. Most notably, this space included:

- new interfaces for musical control and artistic expression;
- wearable interfaces and mobile applications leveraging on biosignals and activity recognition;
- games with new forms of interaction (e.g. multimodal interaction, biosignals, novel sensors);
- application support for customisation-driven interaction using data;
- collaborative authoring applications which build upon the community affordances of the web.

These UCD actions led to a better understanding of the perceptions, expectations and challenges around RAPID-MIX technologies. For instance, we have found that many participants in our UCD actions struggled to grasp fundamental concepts of machine learning—e.g. classifiers, labels, numerical data goes in, and classes come out—even when users were professional developers who were provided with documentation.

The effort of intertwining several rounds of rapid prototyping with user engagement uncovered design problems and specific technical challenges for the RAPID-MIX consortium. For instance, some of the IML technologies such as XMM were favoured over classifiers of current market products for the classification of dynamic body gestures, but seen as more challenging to use. Wekinator was perceived as very user-friendly for beginners and designers with few programming skills and was used to build two prototypes. For a few participants, Wekinator was considered limited for temporal data and recognition of continuous gestures. Overall, we found that there was a general lack of understanding and adoption of RAPID-MX ML technologies.

These UCD actions also led us to identify requirements which had not been anticipated (e.g. preferred environments for integration). They also informed us about how to prioritise our API deployment targets, about some of the trade-offs involved in supporting them (C++ binaries for compatibility with different target environments vs the low appeal for these users to engage directly with), and of potential workarounds to make the design of the future toolkit useful for these users.

Furthermore, these insights also contributed to the formulation of design guidelines for the RAPID-MIX API, which the next chapter will examine. These guidelines embodied RAPID-MIX stakeholders' perspectives about the 'right' set of toolkit features to build, based on the results of UCD actions, expert evaluation of prototypes, and activities of the SIGs.

Early UCD actions also pushed for the identification of subsequent research needs. Particularly, they opened new research questions and more fine-grained UCD goals concerning the need for further efforts in understanding issues and factors related to the difficulties in the adoption of machine learning, and scoping and better understanding our users through subsequent UCD actions. More focused research questions could be asked, such as:

- Specifically, who are the developers using machine learning and what are they needs?
- In what ways should we design machine learning infrastructural software tools which facilitate their adoption for developers?

All considered, the UCD actions of the first action research cycle led to useful insights and more focused questions.

Everything should be made as simple as possible, but not simpler

— Albert Einstein

Chapter 5

Second Action Research Cycle: Action Design Research

The second action research cycle began shortly after the end of the first year of RAPID-MIX. For this cycle I adopted an Action Design Research (Sein et al., 2011) approach. The first stage of Action Design Research consists of the formulation of the research problem with the principles of *Practice-Based Research* and *Theory-Ingrained Artefact* (Section 5.2). Subsequently, this chapter examines the research activities and insights of the *Building, Intervention and Evaluation* stage. In Section 5.4, I examine early activities including the concurrent prototyping and evaluation, expert evaluation of a second-generation prototype, and the collaborative synthesis of the RAPID-MIX API design guidelines. In Section 5.6.1, I provide a technical overview and analysis of the RAPID-MIX API as the design artefact, including the initial development up to its version 1.0 release, highlighting my contributions to the artefact. In Sections 5.7.1, 5.7.3 and 5.8, I examine user evaluation activities that interwove with the RAPID-MIX API development process.

5.1 Introduction to the Second Action Research cycle

By the time the deliverables *D2.1 UCD Methodology* (Bernardo et al., 2015) and *D2.2. Design guidelines for prototyping* (Bevilacqua et al., 2015) had been written—within the first action research cycle, months four and eight of RAPID-MIX, respectively—my research did not have a theoretical grounding *per se.* My collaboration in the project had begun with a "theory-free action learning" stance, as conceded by Davison et al. (2004). However, the threat of "irrelevant subject failure"—conducting a study that is not relevant to the research community (Avison et al., 2001)—hovered over my efforts of finding a suitable HCI theory. Although the theoretical corpus of HCI contemplates situated and grounded theories (Winograd and Flores, 1987; Bodker, 2006; Dix, 2012), they are mostly related to cognitive science theories of action.

These difficulties pushed me to broaden the scope of my theoretical exploration and to seek a theory which I could apply to guide practice and contribute relevant research within the RAPID-MIX innovation context.

I developed a parallel literature review in Innovation Studies, motivated by the general goals of RAPID-MIX in technology transfer and innovation, and my personal interest and background in this field. Around the beginning of the second action research cycle, I discovered von Hippel's User Innovation theory (von Hippel, 2005). I found an interesting alignment between the theoretical constructs of this theory and the practical goals of the RAPID-MIX work package 2 (*WP2 UCD Methodology*). In particular, *Lead Users*, *Sticky Information* and *User-Innovation Toolkits*—the core constructs of von Hippel's User Innovation theory—appeared to accommodate well the theoretical contributions of design research in human-centred machine learning—e.g. the design space for end-user interaction with ML of Amershi (2012) and design principles abstracted from empirical studies more focused on usability of ML systems—as well as the 'nature' of machine learning itself.

5.2 The *Problem Formulation* stage

In this section, I employ the principles of *Practice-Based Research* and *Theory-Ingrained Artefact* and break down problem formulation into a set of sub-tasks. I begin by framing the problem and identifying the research opportunity in Section 5.2.1. In Section 5.2.2, I generalise the problem by casting it as an instance of a class of problems. In Section 5.2.3, I identify the theoretical bases contributing to theoretically frame the problem that my thesis pursues. In Section 5.2.4, I identify the prior technology advances at the beginning of this research iteration in RAPID-MIX.

5.2.1 Framing the anticipated problem

According to Sein et al. (2011, p. 40), the principle of *Practice-Based Research* entails that the research effort should be triggered by an immediate or anticipated problem perceived by organisational participants and then framed by the researcher. Over the years, the RAPID-MIX consortium gathered a high level of knowledge spanning many different domains: Human-Computer Interaction and user-centred design, digital signal processing and machine learning, software engineering, biosignals and sensors for expressive interaction, and technology transfer and innovation. The RAPID-MIX consortium also gathered a portfolio of multimodal, interactive and expressive technologies (Section 1.2.2), including tools that provided good support for creative exploration and rapid prototyping.

The problem which triggered my research was motivated by the lack of tools that could support both rapid prototyping and commercial product development effectively. This was mainly associated with the complexity, low usability, fragmentation and lack of interoperability of pre-existing RAPID-MIX tools. Preliminary research (Section 4.6) indicated that, central to RAPID-MIX consortium's goals, was the creation of new developer tools for integrating multimodal data sources, real-time data flows, digital signal and machine learning processing, and multimedia outputs into client software applications. These tools were aimed at developers, designers, academics and artists, working with technology for creative applications and industries. These profiles were characterised by a range of variable programming proficiency, lack of expertise in machine learning, and different production needs—i.e. ranging between creative exploration, rapid prototyping, and commercial application development, in the areas of music technology, game development and quantified self—as well as different production and deployment environments (i.e. development platforms and target devices).

The problem which triggered my research opportunity, as anticipated by the RAPID-MIX consortium, could be framed, therefore, as the application of a user-centred design process, with users of the above-mentioned profiles, to the creation of a new software development toolkit, the RAPID-MIX API, based on the integration of a portfolio of disparate, broad-scope, general-purpose background RAPID-MIX technologies (see Section 1.2.2). Specifically, this toolkit would consist of infrastructural software and developer tools for rapid prototyping and commercial development of creative applications integrating signal processing and machine learning for multimodal data and expressive interaction. The process of applying user-centred design to the creation of the RAPID-MIX API would entail understanding what to build, and how to build the infrastructural software to be useful, usable and "satisficing" to users aligned with the target user profiles. This would in turn entail understanding the needs, goals and values of users of the toolkit, and exploring the design space of background RAPID-MIX technologies and future integrations.

I apply the principle of *Practice-Inspired Research* (Sein et al., 2011) to approach the task of designing the RAPID-MIX API as a field problem which can provide an opportunity to produce scholarly design knowledge. In the framing supported by these two principles, the RAPID-MIX API becomes an 'ensemble' IT artefact (Sein et al., 2011), a bundle of information technology comprised of the software forged in RAPID-MIX. The RAPID-MIX API becomes an "emergent thing" (Sein et al., 2011) that is shaped by the organisational structure and technological features of RAPID-MIX consortium—i.e. background technologies that were selected and from which the design process departed, and which made their way into the final design, the roles of the consortium stakeholders and how their design decisions determined or influenced the software components, structure, form, presentation, communication about the artefact. The RAPID-MIX API becomes an artefact that is shaped not only by one API designer, but by theory and assumptions, needs, goals and values of the RAPID-MIX consortium stakeholders and end-user communities, and their interactions with RAPID-MIX technologies.

5.2.2 Casting the problem as an instance of a problems class

Sein et al. (2011, p. 40) suggest an initial abstraction step in the problem formulation stage in order to articulate the problem with existing theories. The problem that my thesis pursued could be considered, first of all, a design problem and an instance of the class of ill-structured problems. *Ill-structured problems* (ISPs) were first formulated by Walter Reitman in the 1960s as "ill-defined problems", and later developed by Allen Newell and Herbert Simon in the 1970s. Simon, who makes the distinction between ill-structured and well-structured problems, considered that ISPs are ill-formulated in certain aspects: for instance, there are no well-defined criteria to test a possible solution, no systematic process to apply the criteria to test the solution, the problem space is not well defined concerning design processes, structures or level of effort, or where the domain knowledge might be difficult to understand because it requires tacit knowledge (Simon, 1973).

As with other design problems, my research problem introduced the opportunity to evolve from a current state to the desired state through design. In the previous section (Section 5.2.1), I used insights from my preliminary research to frame the problem concretely under the perspective of design practice within the RAPID-MIX project setting and characterisation of the departure state. Abstracting away the details of the previous section, the problem comprised the application of a user-centred design process to the creation of a new developer toolkit comprising infrastructural software for the application real-time machine learning, based on the integration of knowledge and a set of disparate, broad-scoped, general-purpose technologies.

The desired state to achieve was to have this toolkit available as an open-source resource which was useful and usable for developers and user communities to use for rapid prototyping and product development with machine learning in the context of creative industries—or, intended to broaden and accelerate innovation with machine learning for users across set of different creative profiles. However, apart from these broad and high-level initial goals and broad constraints in terms of technologies, there were no clear goals concerning, for instance, what constituent technologies the toolkit should incorporate, or what would make the toolkit user-friendly or easy to use.

The problem of designing such a toolkit posed complex and ill-structured, considering both the theoretical gaps and the setting in which my research developed. There were no clear initial constraints or guidelines to the design process, or evaluation criteria or specifications for the desired end state, etc. All of these had to be determined while devising a solution the problem. As Buchanan (1992) puts it, providing a solution for such a problem, firstly, entails setting the problem and providing a detailed description; then, the various possible solutions must be considered, and the constituent elements gathered. Hopefully, this leads to the gradual emergence of an optimal solution as an outcome of a continual discussion, argument and critical judgement between the participants of the process (Buchanan, 1992).

5.2.3 Contributing theoretical bases

In this work, I employ the generative power of theory in its synthetic form (Dix, 2012)—i.e. applying theory to inform design. This involves the application of theoretical knowledge for problem setting and problem solving, in order to arrive at the desired design outcome. In this section, I explain how I articulate these theoretical constructs to formulate my research problem and working hypothesis, using the principle of *Theory-Ingrained Artefact* from the *Problem Formulation* stage of ADR (Sein et al., 2011).

Here I introduce the main theoretical bases that ground my research. The first is User Innovation theory (von Hippel, 2005), which has been reviewed in Section 2.2.1. Other theoretical bases contributing to my research are design abstractions and principles originating from the HCI domain, specifically from Human-Centred Machine Learning. These theoretical constructs have been generalised from the analysis of different implementations of IML in different domains, and from the articulation of the emerging design principles with HCI theory. These contributions, which I have previously reviewed in 2.1.2.2, consist of Amershi's (2011) design space for end-user interaction with machine learning (EUI-ML), and Dudley and Kristensson's (2018) design space for comprehensive interactive machine learning (C-IML).

According to both Amershi (2012) and Dudley and Kristensson (2018), there have been a general lack of consolidated design principles in HCML. Similarly, in the domain of User Innovation, Jeppesen and Frederiksen (2006) claimed that there was a lack of principles for structuring user-innovation toolkits. There is a notable convergence in these theoretical bases, which encouraged me to inform the design and guide the evaluation of RAPID-MIX API as a *Theory-Ingrained Artefact*, thus complying with the principles of ADR postulated by Sein et al. (2011).

I apply the principle of *Theory-Ingrained Artefact* to frame the RAPID-MIX API as an artefact that embodies User Innovation theory; I employ User Innovation theory to analyse the different iterations to plan user-centred design, and evaluation interventions to inform design of the RAPID-MIX API. I am applying User Innovation theory to drive design exploration and synthesis—i.e. using theory backwards to ask, "I want this to happen, what should I do to make it happen?" (Dix, 2008). More particularly, if we want to accelerate and broaden user innovation with machine learning, how does User Innovation theory suggest machine learning tools should be designed?

Amershi's (2011) design space for end-user interaction with machine learning aims to provide a framework for the identification of new design challenges, opportunities, and to support systematic approaches to systems design and analysis. By directing attention to the design factors and design dimensions that should be considered, Amershi's (2011) design space can help to raise questions, understand and discuss the design of toolkits for rapid prototyping IML systems.

Following an analogue line of thought, we can bring forward an alternative interpretation of Dudley and Kristensson's (2018) C-IML design space which extends beyond the original intent. An alternative examination of this design space's canonical elements and model-building sub-tasks is as determinants for establishing design features of toolkits for building IML applications. It can be used in the analysis of client-application requirements to inform the design of the ML developer tools.

I hypothesise that we may use Amershi's (2011) and Dudley and Kristensson's (2018) design spaces to understand better the design of ML developer tools—in particular, the design of the RAPID-MIX

API—and the needs of developers building IML systems. Amershi's and Dudley and Kristensson's design spaces may be combined to complement each other, and articulated within the taxonomy of User Innovation theory. With this, we may arrive at an adequate working theory to explore, inform and analyse the design of a toolkit for user innovation with interactive machine learning. User Innovation and the HCML design spaces will be employed therefore as working theory which may provide an effective way to guide method to obtain insight and to understand the nature of the problem of designing interactive machine learning for toolkits for user innovation. It may help, as well, to understand and work more effectively with users in the context of RAPID-MIX, to design and run studies and analyse their outcomes.

5.2.4 Brief analysis of prior technology advances

The prior technology advances in my research problem concern two sets of RAPID-MIX technologies. The first set comprises elements of the portfolio of background technologies for multimodal and expressive interaction that the RAPID-MIX consortium developed and accumulated over the years. These elements are described in Section 1.2.2. The second set of technologies comprises the first generation of prototypes that were created in parallel prototyping activities during the first action research cycle.

According to MacLean et al. (1991, p. 248) "creating the design space, as well as creating the artefact, is a discovery process; it is one and the same discovery process". Gaver (2012) presents the notion of design artefacts embodying a conceptual rationale (e.g. theory, a novel technology, a user need) as points in a design space—or, collectively creating the *space of possible designs* of a certain domain. In the formal use of a design space, dimensions are useful, for instance, for exposing design decisions and how the different possible solutions relate; they can also show areas of the design space that may be considered more appealing or under-explored (Card et al., 1990; MacLean et al., 1991). Dimensions can and should be extracted, optimally in a consensual and collaborative manner, and may be given different levels of significance and priority (MacNeil et al., 2017).

Parallel prototyping and evaluation activities involved a significant amount of effort between all RAPID-MIX partners, including the production of the actual prototypes, their evaluation and documentation. The documentation for each prototype included its design rationale—concerning the lead question, context-of-use and use cases—and technical description, concerning the functionality and assemblage of RAPID-MIX technologies or system architecture. Prototypes were also documented as video-prototypes (Mackay, 2002), which circulated the RAPID-MIX consortium, and in some cases, were used to collect feedback from potential end users.

In RAPID-MIX, at the time of the writing of the official deliverables for work package 3 (*Agile Prototyping*)—*D3.1 Early Prototypes V1* (September 2015) and *D3.2 Early Prototypes V2* (April 2016), to which I also contributed—there was no explicit reference to the notion of design space or to

# Name	Users														
	Developers				Designers				Artists			Academics			End Consumers
	Web	Mobile	Games	Audio	Creative Coders	Hackers	Makers	Sound	Music	Visual	Performance	Researchers	Teachers	Students	
1 FreeMix: Content-based Interaction with Recorded Audio								■	■		■				■
2 Wearable Musical Instruments		■		■	■	■	■		■		■				■
3 Multimodal Feature Extraction & Exploration Toolkit		■	■	■	■	■	■	■				■	■	■	
4 Multimodal Soundwalks					■	■		■							■
5 Online repository with IML capabilities	■	■	■	■	■	■			■		■	■	■	■	
6 Open Biosignal Repository	■											■	■	■	
7 BITalino & OpenSignals (r)evolution	■	■				■	■				■	■	■	■	■
8 RAPID-MIX "sharer" prototype / example / technology	■	■	■	■	■	■							■	■	

# Name	Use Case								
	Biosignals in Context	Hardware Processing	Music Augmentation	Collaborative Web Authoring Tools	Type of Prototype				
					Proof-of-Concept	Technology Integration	Technological Brick	API Demonstrator	Product
1 FreeMix: Content-based Interaction with Recorded Audio			■			■			
2 Wearable Musical Instruments		■	■		■				■
3 Multimodal Feature Extraction & Exploration Toolkit	■				■	■			
4 Multimodal Soundwalks			■	■					■
5 Online repository with IML capabilities				■		■			
6 Open Biosignal Repository	■			■		■			
7 BITalino & OpenSignals (r)evolution	■					■			■
8 RAPID-MIX "sharer" prototype / example / technology				■	■				

# Name	Component Technologies																		
	Sensors Hardware				Feature Extraction				Real-time Audio				Interactive Machine Learning				Online Repositories Collaborative Communities		
	BITalino (PLUX)	OpenSignals (PLUX)	r-IoT (IRCAM)	*3rd-party*	Essentia (UPF)	Weki Input Helper (GS)	Maximilian (GS)	*3rd-party*	Maximilian (GS)	IAE (IRCAM)	CoSiMa (IRCAM)	*3rd-party*	GF (IRCAM)	GVF (GS/IRCAM)	XMM (IRCAM)	Wekinator (GS)	Repovizz (UPF)	Freesound (UPF)	CodeCircle (GS)
1 FreeMix: Content-based Interaction with Recorded Audio	■				■					■	■				■			■	
2 Wearable Musical Instruments				■					■							■			
3 Multimodal Feature Extraction & Exploration Toolkit	■	■					■									■			
4 Multimodal Soundwalks				■							■								
5 Online repository with IML capabilities																■	■		
6 Open Biosignal Repository	■	■															■		
7 BITalino & OpenSignals (r)evolution	■	■															■		
8 RAPID-MIX "sharer" prototype / example / technology																■			

# Name	Platform						Authors							
	Web (server-side)	Web (client-side)	Mobile	Embedded	Desktop	*3rd-party*	GS	IRCAM	UPF	ROLI	Orbe	PLUX	Audiogaming	Reactable Systems
1 FreeMix: Content-based Interaction with Recorded Audio		■	■		■			■	■					
2 Wearable Musical Instruments			■	■			■							
3 Multimodal Feature Extraction & Exploration Toolkit			■	■	■		■							
4 Multimodal Soundwalks	■	■	■					■			■			
5 Online repository with IML capabilities	■				■		■		■					
6 Open Biosignal Repository	■	■							■			■		
7 BITalino & OpenSignals (r)evolution		■	■		■							■		
8 RAPID-MIX "sharer" prototype / example / technology	■				■		■							

Table 5.1: First generation of RAPID-MIX prototypes

its use as a conceptual tool for describing and exploring the design possibilities. Nevertheless, a set of dimensions for high-level characterisation of the prototypes emerged informally and organically which included:

- *Context/Use Case* – characterised the prototype according to a high-level description of context-of-use, scenario, use case, groups of users, or type of functional artefact.
- *Currently used technologies* – prototype component technologies which included elements from the joint innovation portfolio of RAPID-MIX, other technologies within the consortium that made their way into the project (e.g. CodeCircle, r-IoT), or third party technologies (e.g. MYO and Leap Motion sensors, development platforms such as Max/MSP, etc.)
- *Potential integration with other technologies* – candidate technologies from the joint innovation portfolio of RAPID-MIX for integration with the current prototype.
- *Platform* – computational environment in which the prototype was integrated or deployed for use—e.g. web (server-side, client side), mobile, desktop, embedded, development frameworks (Max/MSP, JUCE, Openframeworks, Unity, etc.).
- *Authors* – members of the RAPID-MIX who produced the prototype.
- *Potential contributors* – members of the RAPID-MIX consortium with potential interest in collaborating on the augmentation, refinement or productization of the prototype.

Because of the informal and *ad hoc* nature of the process with which these dimensions were formulated—mainly to document the parallel prototyping activities in RAPID-MIX and provide informal guidance to the process—there was no attempt to provide for the completeness of the design space (Card et al., 1990). These dimensions also have some issues, such as the *Context/Use* dimension aggregating too many important attributes for the characterisation of a prototype. Because of the relevance of these attributes to the design problem, this dimension could be most usefully segregated into the subcategories it represents. *Currently-used technologies* is a also an important dimension because it can illustrate combinations of technologies and articulate design directions which were favoured and positively selected by the RAPID-MIX stakeholders. *Platform* is another important dimension given that it can illustrate how the design decisions related the different possibilities in term of deployment environment. The attribute represented by this dimension was also valued by users in UCD actions, specifically concerning the provision of cross-platform capabilities for the RAPID-MIX API. *Authors* and *Potential contributors* are dimensions which can show the stakeholder engagement and cross-pollination within RAPID-MIX, and were primarily important for management and coordination of the project, but which can also show how the ensemble artefact was built.

The classification of prototypes with these dimensions was useful for the operational needs of RAPID-MIX at the time. However, in order to support a richer, broader, and consistent analysis across the two generations of RAPID-MIX prototypes (tables 5.1 and 5.4), I expanded the initial set of dimensions with new, more granular dimensions and a different visual organisation. I extracted the *Users* dimensions according to the categories of the composite user profile that were identified in previous studies in the last chapter (see 4.6). *Use Case* is broken down into the themes that were identified by RAPID-MIX stakeholders in the co-design sessions (Section 4.3) and one significant theme was extracted from prototype descriptions by the prototypes authors (*D3.1 Early Prototypes V1* and *D3.2 Early Prototypes V2*). A sub-dimension *Type of Prototype* was inspired by the taxonomy of prototypes role in relation to design space and design rationale of Chu et al. (2013). *Component Technologies* are elements from the joint innovation portfolio of RAPID-MIX aggregated according to the themes of interest of the SIGs, with the addition of *3rd-party* dimension to each sub-category to capture and aggregate the use of external technologies to RAPID-MIX in the prototypes. The *Authors* dimension remained unchanged. For the *Platform* dimension I extracted the following sub-categories more clearly —*Web(server-side)*, *Web(client-side)*, *Mobile*, *Embedded*, *Desktop* and *3rd-party.*

Table 5.1 below is populated with instances corresponding to first-generation RAPID-MIX prototypes. These prototypes were focused mostly on the integration of technologies (6 of 8 prototypes) from different RAPID-MIX consortium stakeholders. There were prototypes which also had more of a proof-of-concept nature (3 of 8 prototypes), while others were early versions of products (3 of 8 prototypes). Notably, with the exception of the MFEET (prototype #3) and BITalino (prototype #8), which were considered toolkits providing elements targeting developers, there was a low occurrence of technological bricks build purposely for developers to use. Instead, most of the first-generation prototypes provided high-level GUIs, which made them suitable for a wider range of users, including developers, designers, artists, academics, and end consumers.

Among the prototypes which also targeted developers, there were rapid prototyping tools (prototype #3, MFEET), complementary tools for storage (prototypes #5, “Online repository with IML”, and #6, “Open Biosignal Repository”), or online collaboration tools (prototype #8, the RAPID-MIX “sharer”). Prototypes embodying the *Collaborative Web-authoring tools* use case were predominant (4 of 8 prototypes). This is consistent with the findings from the co-design sessions (section 4.3) with regards to the convergent interests of RAPID-MIX stakeholders in using the Web as medium for tool delivery and production. Additionally, some prototypes provided early versions of products, thus targeting end consumers (e.g. prototypes #2 “Wearable Music Instruments”, #4 “Multimodal Soundwalks” and #7 “BITalino OpenSignals (r)evolution”). As table 5.1 shows, technologies from different *Component Technologies* categories were explored across prototypes with slightly higher incidence in the *Sensors & Hardware* and *Interactive Machine Learning* dimensions.

There was a balanced co-occurrence of deployment environments for first-generation prototypes, concerning *Web* and *Desktop*, as characterised by *Platform* dimension. There was, nevertheless, a significant overlap of these dimensions with *Mobile* (e.g. prototypes #1 “FreeMix” and #4 “Multimodal Soundwalks"), which can be explained by the cross-platform capabilities of some of the component technologies—e.g. Maximilian was written in C++, and as such, supports applications for both mobile and desktop deployment—and also by some of the web-based technologies (e.g. CoSiMa), which were usable in both desktop and mobile browsers.

5.3 The *Building, Intervention and Evaluation* stage

According to the ADR methodology (Sein et al., 2011), the stage of *Building, Intervention and Evaluation (BIE)* interweaves the building of the design artefact with interventions within the research client organisation and evaluation activities. As previously described in Section 3.5, this stage requires an initial identification of the locus of innovation and target of knowledge creation from the ADR continuum. This entails choosing a research focus from between the end points of the continuum, the *IT-dominant BIE*, which emphasises the creation of technological artefact, and the *Organisation-dominant BIE*, which emphasises the intervention within the organisation (Figure 3.4, pg. 65).

5.3.1 Identification of the locus of innovation and target of knowledge creation

My approach to the BIE stage takes an intermediate positioning in the ADR BIE continuum, balanced between artefact and intervention. My research puts slightly more emphasis on intervention in RAPID-MIX for the creation and improvement of the design artefact—i.e. applying User-Centred Design Actions to inform the creation and improvement of RAPID-MIX as the outcome of a participatory and user-centred process. Specifically, as I will show throughout this chapter, my research contributed to challenge "organizational participants' existing ideas and assumptions" (Sein et al., 2011, p. 42) with an understanding of the characteristics of end users and contexts-of-use, obtained through UCD actions for the assessment of the RAPID-MIX API and related artefacts. In this regard, my work also contributes more generally with knowledge about the application of UCD in RAPID-MIX. Additionally, my research also focuses on capturing design knowledge from the effort of the RAPID-MIX consortium in order to create an innovative design for the RAPID-MIX API, as an easy-to-use developer toolkit for rapid prototyping with interactive machine learning.

ADR prescribes three principles for the BIE stage. By adhering to the first principle *Reciprocal Shaping*—i.e. "influences are mutually exerted by [...] the IT artefact and the organisational context" (Sein et al., 2011, p. 43)—I will illustrate in this action research cycle how the RAPID-MIX API was shaped by the organisational context of RAPID-MIX, including RAPID-MIX stakeholders and end users inside and outside the consortium. This includes an overview of parallel prototyping activities, a collaborative synthesis of the RAPID-MIX API design guidelines, initial development, and evaluation interventions that interwove with this process. Contrastingly, I will show how the design process of the RAPID-MIX API also shaped the assumptions and perceptions of stakeholders and end users. Using the principle of *Mutually Influential Roles*—i.e. "mutual learning among the different project participants" (Sein et al., 2011, p. 43)—I illustrate how different stakeholders have contributed to and learned in the process of development of the RAPID-MIX API. This includes showing how the insights of the UCD process contributed to the design of the API, and how some of the design decisions for the RAPID-MIX API were established within the consortium.

By adhering to the principle of *Authentic and Concurrent Evaluation*—i.e. "evaluation is not a separate stage of the research process that follows building [...] it is on-going" (Sein et al., 2011, p. 43)—I show how in this cycle both the research problem that was previously formulated and the artefact were continuously evaluated, with particular consideration given to the design principles that it embodies as an ensemble and IT-ingrained artefact. The RAPID-MIX API was evaluated using UCD actions with different user groups that employed different techniques. These UCD actions will be examined in Sections 5.7.1 and 5.7.1.

5.3.2 Action planning

UCD actions planned within RAPID-MIX in the period of the second action research cycle included the evaluation of second-generation prototypes to inform the production of the RAPID-MIX API guidelines. After the arrival of Michael Zbyszyński to the RAPID-MIX project, who took on the role of lead developer of the RAPID-MIX API, in late 2015, and after the period that encompasses early development and first releases of the constituent elements of the RAPID-MIX API, there was a need to prepare an evaluation of the toolkit.

In the period leading to milestone M2 ("RAPID-MIX prototypes deployed in relevant environments"), the efforts to formulate the definition of design guidelines for the RAPID-MIX API began with the organisation and formation of Special Interest Groups (SIGs) within the RAPID-MIX consortium (UPF, Barcelona, November 2-4th, 2015). Different RAPID-MIX stakeholders joined the SIGs, which focused on themes leading to the production of the second-generation prototypes and informed the RAPID-MIX API design guidelines. These themes and constituent members of the SIGs were as follow:

a) *Interactive Machine Learning* – F. Bevilacqua, R. Fiebrink, F. Bernardo, M. Zbyszyński, J. Larralde;

b) *Feature Extraction* – M. Grierson, P. Papiotis, R. Fiebrink, F. Bernardo, M. Zbyszyński;

c) *Real-time Audio Synthesis* – M. Grierson, N. Schnell, J.B. Thiebaut, J. Larralde;

d) *Online Repositories and Collaborative Communities* – P. Papiotis, C. Julia, X. Boissarie, A. Bucci, N. Schnell, J. Larralde;

e) *Sensors and Hardware platforms* – H. Silva, E. Flety, F. Bevilacqua, M. Grierson, M. Zbyszyński;

I joined the *Interactive Machine Learning* and the *Feature Extraction* SIGs to assist and contribute to the production of the second-generation prototypes and the guidelines under these themes. I was involved in the activities which led to the production of *D2.3 Design guidelines for the RAPID-MIX API* (Mealla et al., 2016)—I wrote the Feature Extraction section, contributed to the writing of different API component sections and was one of the internal reviewers of the document. These documents have been published online and are available to the general public. Included in these activities was a UCD action I planned more independently:

- an expert evaluation of the *Multimodal Feature Extraction and Exploration Toolkit* (MFEET) prototype (Goldsmiths, July 11-13th, 2016, month 17) (Section 5.4), as part of the production and concurrent evaluation of second-generation prototypes.

After the release of the RAPID-MIX API, my goal was to evaluate different RAPID-MIX API subsets with different user groups in the wild. In subsequent UCD actions, I intended to investigate specifically: a) the needs, goals and values of two different target user groups; b) how they used the RAPID-MIX API and what for; c) usability issues using the RAPID-MIX API; and d) other unexpected findings from which could suggest new features or re-design in the RAPID-MIX API. To achieve these goals I planned the following UCD actions:

- JUCE Machine Learning Hackathon with audio software developers (London, December 16, 2016, month 23) (section 5.7.1).
- Two-week summer workshop with creative developers at ENTERFACE'17 (Porto, July 3–15, 2016, month 29) (section 5.7.3).

For the planning of the JUCE ML Hackathon, I collaborated with Fabian Renn-Gilles (JUCE lead developer at ROLI), who led the UCD action, Sebastian Mealla (MTG/UPF), and Rebecca Fiebrink (Goldsmiths). I led the eNTERFACE'17 workshop, which was mostly planned by me with the help of Rebecca Fiebrink. Nearer the date of the eNTERFACE'17 workshop, the planning received input from Atau Tanaka (Goldsmiths), Sebastian Mealla and Panos Papiotis (MTG/UPF), and Joseph Larralde (IRCAM). Figure 5.1 (p. 136) shows a diagram with these UCD actions referenced within the map of RAPID-MIX Action Research cycles. I revised the initial action research plan and map to deprecate the third action research cycle and instead include one single encompassing action design research cycle. The specific sections that report these UCD actions provide additional details about their planning.

These UCD actions, which had my direct involvement and intervention, were part of a broader set of UCD actions which were deployed during this period. Other UCD actions that were deployed without my involvement included, for instance, specific prototype evaluations with users and the

Sonar Innovation Challenge, which was a large-scale UCD action organised by MTG/UFP and deployed in June 15th-17th (month 17) in Barcelona. These UCD actions also contributed with insights for the production of the design guidelines and were documented in *D2.3 Design guidelines for the RAPID-MIX API* (Mealla et al., 2016). The following sections provide detail on the different UCD actions that I have planned.

5.4 Second-generation parallel prototyping and evaluation

In this section, I provide a high-level overview of the second-generation parallel prototyping and evaluation stage in RAPID-MIX. Parallel prototyping and evaluation aimed to engage the different RAPID-MIX stakeholders in learning about the possibilities of the different RAPID-MIX technologies and exploring the possible design directions for early prototypes of MIX products and RAPID-MIX API. This stage was important in the overall design process because it contributed to the production of its design guidelines. I illustrate the overall process with the description and analysis of the MFEET prototype expert evaluation, as one instance of the concurrent prototyping process in which I directly intervened.

		Users															Use Case										Component Technologies																				Platform						Authors							
		Developers				Designers				Artists			Academics			End Consumers	Biosignals in Context	Hardware Processing	Music Augmentation	Collaborative Web Authoring Tools	Interactive Machine Learning and Data-driven Custom Interaction	Type of Prototype					Sensors Hardware				Feature Extraction					Real-time Audio				Interactive Machine Learning				Online Repositories Collaborative Communities																
#	Name	Web	Mobile	Games	Audio	Creative Coders	Hackers	Makers	Sound	Music	Visual	Performance	Researchers	Teachers	Students							Proof-of-Concept	Technology Integration	Technological Brick	API Demonstrator	Product	BITalino (PLUX)	OpenSignals (PLUX)	r-IoT (IRCAM)	*3rd-party*	Essentia (UPF)	PiPo (IRCAM)	Weki Input Helper (GS)	Maximilian (GS)	*3rd-party*	Maximilian (GS)	IAE (IRCAM)	CoSiMa (IRCAM)	*3rd-party*	GF (IRCAM)	GVF (GS/IRCAM)	XMM (IRCAM)	Wekinator (GS)	Repovizz (UPF)	Freesound (UPF)	CodeCircle (GS)	Web (server-side)	Web (client-side)	Mobile	Embedded	Desktop	*3rd-party*	GS	IRCAM	UPF	ROLI	Orbe	PLUX	Audiogaming	Reactable Systems
1	FreeMix: Content-based Interaction with Recorded Audio v2								■	■		■				■			■				■				■				■	■					■	░				■		■	■			░	■		■			■	■					
2	Wearable Musical Instruments v2		■		■	■	■	■		■		■				■		■	■			■				■				■						■							■						■	■			■							
3	Multimodal Feature Extraction & Exploration Toolkit v2		■	■	■	■	■	■	■	■	■	■	■	■	■		■				■	■	■				■			■			■	■									■						■	■	■		■							
4	ORBE-MIX: Multimodal Soundwalks v2					■	■		■					■		■			■	■						*Orbe*				■		■					■	■				■					■	■	■					■			■			
5	*Online repository with IML capabilities*	■	■	■	■	■	■			■		■	■	■	■					■	■		■																				■	■			■				■		■		■					
6	*Open Biosignal Repository*	■											■	■	■		■		■				■				■	■																■			■	■							■			■		
5	(Bio)Signals in context	■	■	■	■	■	■			■		■	■	■	■		■			■	■					RS				■													■	■									■		■			■		■
6	BITalino & OpenSignals (r)evolution	■	■	■		■	■	■				■	■	■	■	■	■						■			■	■	■																■				■	■		■							■		
8	*RAPID-MIX "sharer" prototype / example / technology*	■	■	■	■	■	■							■	■					■		■																					■				■				■		■							
7	Multimodal recognition					■	■	■	■	■		■				■			■		■	■							■										■			■										■		■	■					
8	WIML: Web Interactive Machine Learning	■	■	■		■	■					■	■	■	■						■	■	■	■		*Orbe*	Sensors&Hardware							FeatureExtraction						■		■					■	■	■					■						
9	MaxiLib.js	■		■	■	■	■	■	■				■	■	■								■	■										■	■											■		■				*emsc*	■							
10	PiPo Plugin in JUCE		■		■																		■	■		*JUCE*															RealtimeAudio								■		■			■		■				
11	BITalino UART to OSC bridge and WIFI transmission																■	■				■	■				■		■			■																		■				■				■		
12	IML-enabled sound design mobile interface				■				■	■		■			■				■		■	■	■			AG				■		■	■						■			■					■	■					■						■	
13	Wekinator JSON exporter	■	■	■	■	■	■	■	■	■	■	■	■	■	■		■	■	■		■		■	■	■	■				■													■				■	■	■	■	■		■							
14	RapidLib C++		■	■	■	■	■						■	■	■		■	■	■		■			■		JUCE																	■				■		■	■	■	■	■							
15	JUCE ML Module		■		■	■	■						■	■	■		■	■	■		■			■	■	JUCE																					■		■	■	■	■	■							
16	RapidLib.js	■	■	■	■	■	■						■	■	■		■		■	■	■			■																			■			■	■	■	■		■	■	■			■				
17	Mano.js	■	■	■	■	■	■						■	■	■		■		■	■	■			■						■		■								■		■					■	■	■		■	■		■						
18	lfo-motion	■	■	■	■	■	■						■	■	■		■		■	■	■			■						■		■																■	■		■	■		■						
19	waves-lfo	■	■	■	■	■	■						■	■	■		■		■	■	■			■						■		■																■	■		■	■		■						

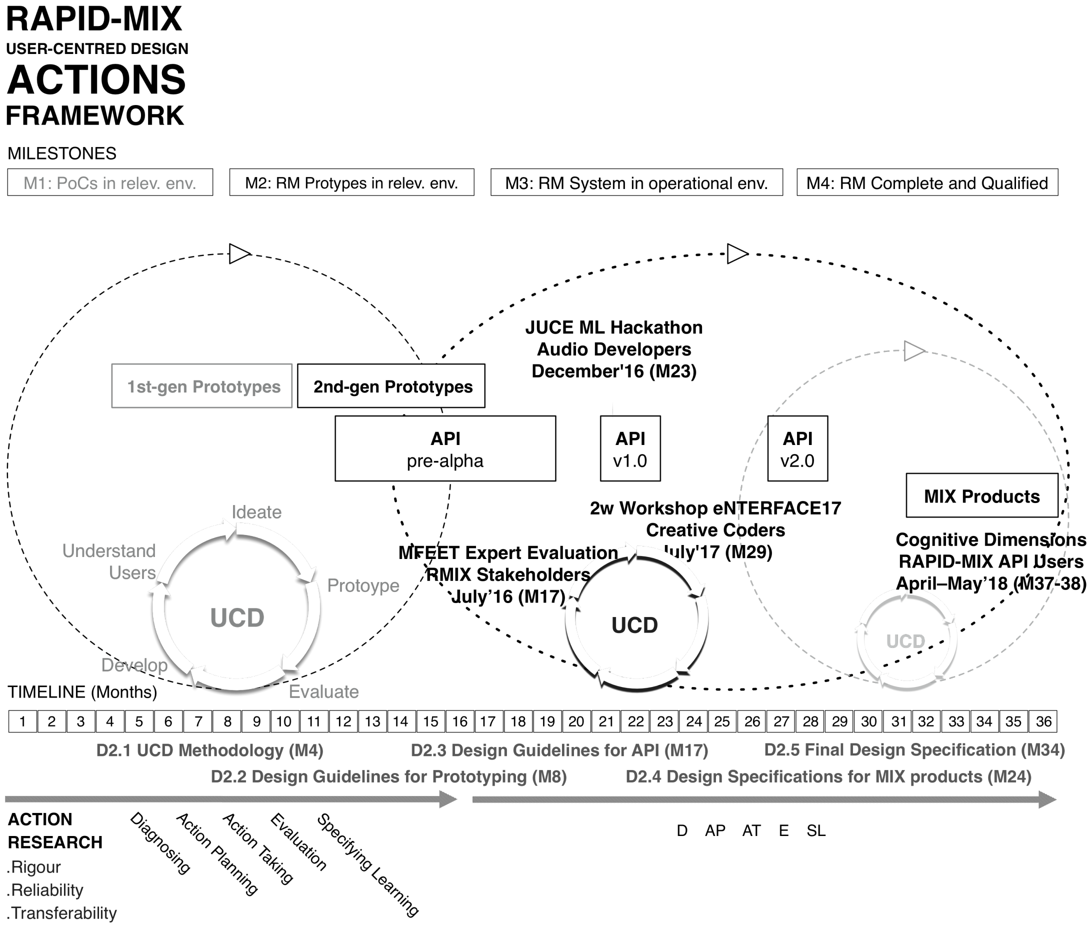


Figure 5.1: UCD Actions of the 2st AR cycle

Second-generation prototypes created within RAPID-MIX during the period of the second action research cycle are shown as instances populating table 5.4. These include augmented or further refined first-generation prototypes (#1 to #4, and #6), augmented background technologies (#9, #10, and #13), and new prototypes (prototypes #5, #7, #8, #11, and #12). Although technological integration still played an important role in second-generation prototypes, there was a concentration of second-generation prototypes around 'building blocks' which targeted explicitly developers of the different domains of interest, and of varying levels of skill and proficiency. This is shown by a meaningful number of artefacts that were produced as technological bricks in this period (prototypes #14 to #20).

Instances of augmented or further refined first-generation prototypes show the evolution of the prototype, both in terms of change of constituent technologies and of deployment environment. In these prototypes, integrated technologies that persisted unchanged are shown in black. New integrations of technology in the prototypes are shown in the color of the *Component Technology* dimension. Technologies or platforms which were discarded from the initial design are shown in grey. For instance, Freemix (prototype #1) evolved by integrating Repovizz and dropping CoSiMa, which determined the fixation of the design around a native mobile implementation, eliminating the design path that comprised web technologies; MFEET (prototype #3) introduced a new sensor bridge for the MYO sensor (3rd-part shown in blue); BITalino & OpenSignals (r)evolution introduced integration Repovizz (shown in blue); ORBE-MIX (prototype #4) introduced XMM integration in the server-side architecture. New prototypes of the second generation are, for example, prototype #5, which was derived from first-generation prototypes that merged (i.e. greyed out entries #5 and #6), and prototypes in the range #7–#12.

These 'building blocks' from table 5.4 show distinguishing features when compared with other second-generation prototypes and first-generation prototypes in table 5.1. For instance, one distinguishing characteristic that may be observed in table 5.4 is the contrasting density of prototypes in the dimensions *Use Case* and *Platform.* As shown by the density in these two dimensions, 'building blocks' open more possibilities for use cases as essential components that enable scaling up a multitude of applications. Furthermore, table 5.4 shows concentration in the dimensions *Interactive Machine Learning* of *Component Technologies* as well as in the use case *Interactive Machine Learning and Data-driven Custom Interaction*, which indicates the fixation of design artefacts along these dimensions. Most notably, these 'building blocks' became the constituent elements of RAPID-MIX API, and will be examined in further detail in Section 5.6.

As part of the parallel prototyping activities of the second action research cycle, I carried out an expert evaluation of the Multimodal Feature Extraction and Exploration Toolkit (MFEET).

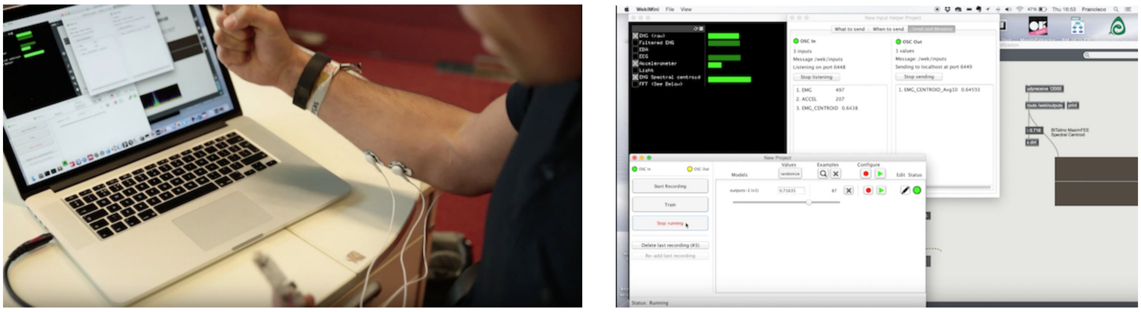
> *The Multimodal Feature Extraction and Exploration Toolkit (MFEET) is a suite of standalone tools that can be connected with each other or with developers' code, which make it possible to experiment with different features and different signal processing techniques very quickly, without programming or worrying about how to interface with hardware. The MFEET assists on the task of determining exactly what information to extract from a raw signal when working with audio signals, physiological signals, or signals from many other types of sensors.*

(Rebecca Fiebrink, excerpt from internal memo with full description of the MFEET, Section E.1, Appendix E).

The MFEET prototype enabled a rapid prototyping approach to feature extraction and machine learning using pipelines with Weki Input Helper and Wekinator as end-user applications with high-level interfaces. My main goal with the evaluation of MFEET prototype was to inform the design guidelines for the RAPID-MIX API, specifically, for the subset of feature extraction and signal processing components. This included obtaining a better understanding about:

a) how expert participants perceived the space of compelling applications that the MFEET could provide for as a rapid prototyping tool.

b) which easy-to-use feature extraction and signal processing functionalities for real-time multimodal data flows expert participants would expect and how they would prioritise them.



(a) (b)

Figure 5.2: Screenshots of the MFEET video prototype featuring a) a custom-built pipeline comprised of BITalino OSC bridge, Weki Input Helper, Wekinator, and Max/MSP, and b) my demonstration of the sonification of forearm muscle activation data captured with the BITalino EMG sensor

c) the general perception of expert participants about MFEET usability, regarding the exploration, configuration of feature extractors and the state of the pipeline, and how this could inform the design of developer tools.

The following section describes the procedure and the results of this UCD action.

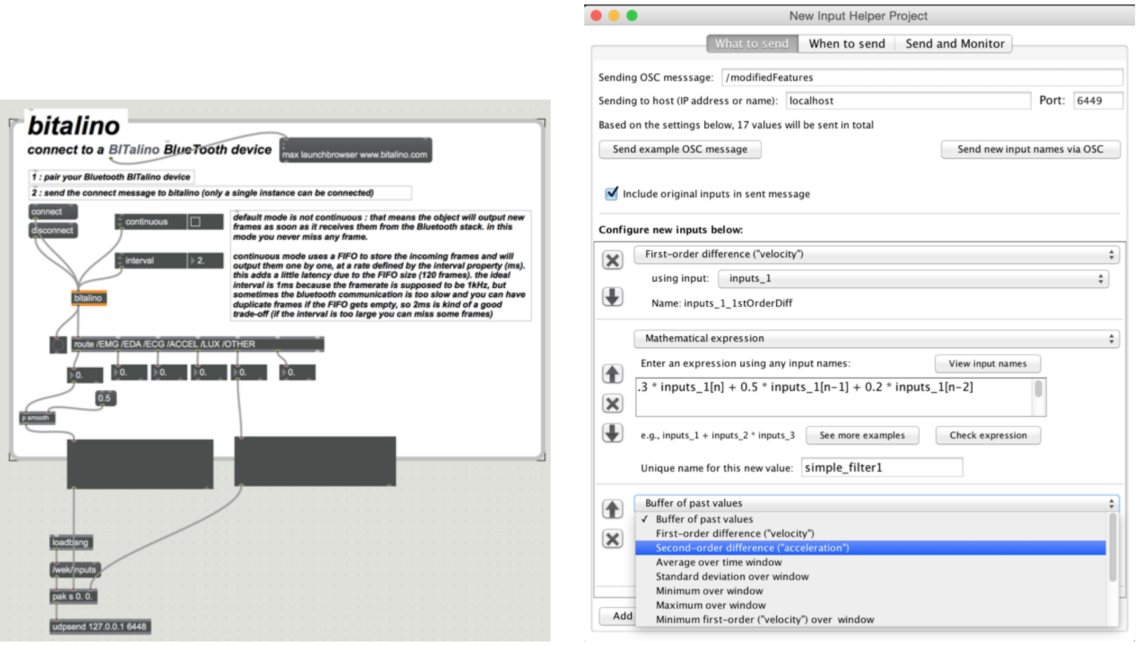
5.4.1 Method

MFEET was a prototype that crossed both the first and second generation of the parallel high-fidelity prototyping—prototype #3 both in table 5.1 and table 5.4.

As with other second-generation prototypes, we created a video prototype (Mackay, 2002), with the technical description and design rationale for MFEET (see Section E.1.2) that circulated internally within the RAPID-MIX consortium. The MFEET video prototype described the MFEET prototype and its design rationale—with Rebecca Fiebrink's voice-off narration. The video prototype was produced by Goldsmiths' team:

- Rebecca Fiebrink authored MFEET, its description and design rationale, and performed, narrated and recorded parts of the video.
- I wrote the Myo sensor bridge[1] and the Max/MSP sonification patch, and built an interactive machine learning pipeline that was used in the video prototype. I also performed and recorded parts of the video, in which I illustrated the setting up and use of a pipeline for the sonification of real-time biosignal data, with BITalino, Weki Input Helper, Wekinator and a sonification patch in Max/MSP (Figure 5.2).
- Steph Horak edited the video.

[1] A sensor bridge can be understood here as a custom software application or component which obtains data from the sensor driver in the operating system, and which encapsulates data streaming into a communication protocol for other applications to consume.



1 : pair your Bluetooth BITalino device

2 : send the connect message to bitalino (only a single instance can be connected)

default mode is not continuous : that means the object will output new frames as soon as it receives them from the Bluetooth stack. in this mode you never miss any frame.

continuous mode uses a FIFO to store the incoming frames and will output them one by one, at a rate defined by the interval property (ms). this adds a little latency due to the FIFO size (120 frames). the ideal interval is 1ms because the framerate is supposed to be 1kHz, but sometimes the bluetooth communication is too slow and you can have duplicate frames if the FIFO gets empty, so 2ms is kind of a good trade-off (if the interval is too large you can miss some frames)

(a)

Sending OSC messsage: /modifiedFeatures

Sending to host (IP address or name): localhost Port: 6449

Based on the settings below, 17 values will be sent in total

☑ Include original inputs in sent message

Configure new inputs below:

First-order difference ("velocity")

using input: inputs_1

Name: inputs_1_1stOrderDiff

Mathematical expression

Enter an expression using any input names:

.3 * inputs_1[n] + 0.5 * inputs_1[n−1] + 0.2 * inputs_1[n−2]

e.g., inputs_1 + inputs_2 * inputs_3

Unique name for this new value: simple_filter1

(b)

Figure 5.3: Screen capture of elements of the MFEET prototype pipeline: a) BITalino MAX/MSP OSC bridge and b) Weki Input Helper

For the assessment of MFEET, I adopted an expert inspection (Preece et al., 2015) with four one-on-one sessions with participants. Given that the prototype originated from Goldsmiths' team, I selected for participants a group of qualified experts from RAPID-MIX consortium stakeholders. The group comprised the following RAPID-MIX consortium stakeholders:

1. Hugo Silva – Expert in biosignals and Chief Innovation Officer at PLUX
2. Frédéric Bevilacqua – IRCAM Sound Music Movement Interaction team leader
3. Amaury La Burthe – Founder and CEO at Audio Gaming
4. Fabian Renn-Gilles – JUCE lead developer at ROLI

I contacted participants and scheduled the expert evaluation sessions for MFEET by email. Given the geographical spread of the participants, three of the sessions were conducted remotely by Skype (participants P01.A, P02.A and P03.A). I did the last MFEET evaluation in a co-located session with participant P04.A at the EAVI lab, in Goldsmiths, University of London. I provided online consent forms to the remote session participants, which required them to read and click a button in order to give their consent. I provided participant P04.A a paper consent form. All participants gave their consent.

I began the procedure debriefing participants about the goals of the session and introducing the components of MFEET (10 minutes). I followed a loose protocol (Section E.2) to walk the expert participants through an MFEET pipeline and facilitate their exploration of the prototype. In the remote sessions, I demonstrated to expert participants the setup of the pipeline and operation with the prototype in a shared screen via Skype. With participant P04.A, I did a walkthrough with Wizard-of-Oz and the participants engaged in open-ended experimentation of the sensor signals and feature extractors provided by MFEET.

I adopted the Wizard-of-Oz technique, a standard evaluation method where users interact with a prototype (with different possible degrees of fidelity) as if they were interacting with a completed product, with a human operator simulating tasks or providing for the shortcomings of the prototype (Norton et al., 2010). In this case, the MFEET was a high-fidelity and functional prototype, and I used Wizard-of-Oz to overcome the constraints of setting up, and as a means for accelerating and guiding the session to its main focus—the expert validation of the prototype, perception of the main usability issues and utility, and space of potential applications.

The procedure included demonstrating how MFEET enabled to set up a pipeline (10 minutes) comprising a sensor bridge—e.g. for sensors such as BITalino, Myo bracelet—Weki Input Helper, Wekinator, and a custom Max/MSP patch that I programmed, which used incoming data through OSC for real-time sonification of the forearm muscle activation data. I chose a different sensor bridge for each pair pf two participants. For participants P01.A and P02.A, I set up Max/MSP running the BITalino object with EMG and 1-channel accelerometer data sent through OSC to Weki input helper for signal processing. For participants P03.A and P04.A, I set up a console application pushing 18 data features from the Myo sensor (3x accelerometer, 3x gyroscope, 4x quaternion, 8x EMG) to Max/MSP, which allowed the selection and visualisation of the raw signals to send to Weki Input Helper.

The remaining pipeline was common to all participants, in terms of constituent components. I used the pipeline to demonstrate the application of feature extraction and signal processing techniques in the interactive machine learning workflow for gesture recognition and sonic interaction (20 minutes). The feature extraction configuration was performed in one of the elements of the pipeline, Weki Input Helper. I focused the demonstration of use on this standalone application, which received data from the sensor bridge, applied a configurable set of feature extractors and signal processing to the data, and pushed results onto the next element of the pipeline. I set up the sensor bridge to send data to Weki Input Helper and used it to apply the moving average to incoming data. I also set up Wekinator to apply classification using the kNN algorithm with the Weki Input Helper features. In the remote sessions, I demonstrated the application of three signal processing techniques which the Weki Input Helper makes available—the pre-defined average over time window, threshold detection with a user-defined constraint, and application of the average over time window using the custom *Mathematical expression* for filters. In the co-located session, I guided participant P04.A through theses tasks, who engaged in exploration and inspection by thinking aloud, as I took notes.

After the exploration of the prototype, I performed semi-structured interviews (30 minutes) with participants. I asked them about their perception and understanding of MFEET, concerning a) the space of compelling applications and tasks which it provided (Topic 4, Section E.2), b) features that

Figure 5.4: Fabian Renn-Gilles (JUCE/ROLI) evaluating the MFEET prototype

should be prioritised to facilitate extraction and application (Topic 5, Section E.2), c) perception of the usability of MFEET concerning the exploration, configuration and state of the pipeline and feature extractors (Topic 6, Section E.2). The screencasts of the remote sessions and the interactions of participant P04.A with MFEET were video recorded and subsequently analysed.

5.4.2 Results

I distilled the results from the four sessions of expert evaluation of the MFEET prototype. I present the findings here organised around the topics of discussion which I defined in the procedure (Section E.2, Appendix E).

1. *Space of compelling applications and utility* – all participants recognised that this prototype fitted in the space of tools for exploration and prototyping of sensor-based signals and feature processing, for pattern recognition based on machine learning. Participants P03.A and P04.A stated that MFEET provided a good way to explore signals which were not intuitive. Participants considered that applications built with MFEET had a high dependency on the scenario of use and technical expertise of the user. Expert participants referred to the space of compelling applications as “wide” (participant P03.A) and “very dependent of the scenario and sensors” (participant P02.A). Participant P01.A considered MFEET target users were non-experts.

   Participants P01.A and P02.A suggested for further development iterations to evolve this prototype to support applications in production—by making it cross-platform and multi-target, i.e. to allow development and deployment on different platforms—while maintaining its affordances for exploration and prototyping. Participants P01.A, P02.A, and P03.A suggested that this pipeline approach could be integrated in different prototyping and programming environments. Environments provided as examples included Max, Pd, OpenFrameworks, JUCE, Unity, Scratch or Lego Mindstorms. A common denominator to a majority of these environments—except OpenFrameworks, which requires textual coding in a high-level flavour of C++—is that they are visual and support graph or data-flow programming; integration of the prototype in these environments would provide more technically novice end-users with a friendlier learning curve.

2. *Feature extractors exploration and configuration* – participants made remarks about positive and negative aspects of feature extraction exploration and configuration functionalities of MFEET, which were concentrated on Weki Input Helper as the component of the pipeline. Positive remarks focused mostly on the usefulness of MFEET as a rapid prototyping and educational tool. Negative remarks highlighted limitations of the provided set of feature extractors in terms of coverage and level of abstraction, and the trade-offs for different user profiles, in particular support for novice users.

   - Positive aspects:

     a) All experts considered the Weki Input Helper useful, and that in general, it provided for the convenience of the user; they considered that Weki Input Helper fills an important gap in the Wekinator toolset. Participants P01.A and P02.A considered that the toolset was very useful for educational activities.

     b) Experts made positive remarks concerning the available selection and organisation of feature extractors in Weki Input Helper. Participants P01.A, P02.A, and P03.A considered that the Weki Input Helper implemented a limited set of features that were very useful for demonstration or educational purposes about signal processing. Participant P04.A found that Weki Input Helper provided for conditioning the signal as he desired, specifically for smoothing the signal and setting the throttling rate correctly for feeding into the machine learning component of the pipeline.

     c) Participant P04.A found the custom mathematical filter notation compensated for some of the shortcomings of the base set of feature extractors and allowed him to attain the operation he was aiming for, applying "average absolute value over time". Participants P01.A, P02.A and P04.A considered the custom mathematical filter notation expression very convenient for the operation of a pipeline.

   - Opportunities for improvement:

     a) There was general agreement that basic features which should be provided off-the-shelf to the user were missing (participants P01.A, P02.A, and P03.A). The following list aggregates suggestions of features to be provided to the user: i) bypass, ii) normalisation, iii) median, iv) envelope, v) modulus, vi) offset, vii) subtraction of the mean/average to compare signals, viii) filters (low pass, high pass and one-pole), ix) accumulator, and x) RMS.

     b) Some participants considered Weki Input Helper's support for creative exploration with feature extractors limited (participants P01.A, P02.A and P03.A). This remark focused on novice users not having access to a set of presets and expert users who would struggled to implement more sophisticated signal conditioning, requiring compromise or workarounds.

i) There were general remarks across experts concerning the use of feature extractors by novice users. Participants considered such users could find feature extractors cryptic or cumbersome (participants P01.A and P03.A). Participant P03.A considered this could dismiss their use altogether. Participant P03.A also considered the custom Mathematical Expression a workaround for the limitations of the feature set that would be difficult to use by such users. Suggestions for improvement of Weki Input Helper focused on the configurability, by providing presets or pre-defined building blocks (participants P01.A, P02.A and P03.A). Suggestions for improvement considered the aggregation and association of the necessary feature extractors to a task (participant P01.A) and black-boxing (participant P02.A) to abstract the complexity of their configuration.

ii) Another aspect mentioned across participants was the need to provide the expert user with new mechanisms for supporting greater flexibility in configuration and extensibility with new features (participants P01.A and P02.A). Participant P02.A suggested including an option for adding new features such as plugins and configurable feature chains which would allow both serial and parallel processing and hierarchies or groups to organise these chains.

c) The interface for feature selection in Weki Input Helper induced two participants in error (participant P04.A). They assumed that the processing operations were happening sequentially across the added feature blocks, and attempted to locate and set input and outputs between the blocks as if they were a chain.

3. *State and configuration of the pipeline* – The configuration of the MFEET prototype depended on a set of independent processes—i.e. sensor bridge, Weki Input Helper, Wekinator, Max/MSP patch–bound by the OSC communication protocol. Participants considered that this MFEET feature provided usability trade-off, that was both useful and limiting. Benefits and caveats are reported as follows:

- Positive aspects:

  a) Participants P01.A and P02.A considered that the interconnection of independent software application processes in a pipeline provided a flexible way to structure a functional, albeit limited environment for signal exploration and feature extraction for interactive machine learning.

  b) Participants P01.A and P02.A considered that MFEET successfully demonstrated how the core aspects of DSP and ML could be applied practically for tasks such as gesture recognition and classification, or music information retrieval. It was considered a useful tool for educational activities such as classes and workshops, where users could be guided through tasks.

c) Participants P04.A considered having the possibility to rename channels throughout the different stages of the pipeline was convenient in supporting the conscious selection of input channels to use (e.g. selecting the specific EMG channels in the Myo which were more relevant to specific gestures).

- Opportunities for improvement:

  a) Despite the benefits previously mentioned, the setup and assembling of the pipeline as a set of interconnected standalone applications were considered to be cumbersome to novice users, particularly without supervision (participants P01.A, P02.A and P04.A). There was a consensus among participants about the challenge of supporting the visibility of the general state of the pipeline, as well as, preventing and recovering from errors in distinct parts of the pipeline assembly. There was also a consensus among expert participants that having the same functionality running in an integrated and unified environment would be more desirable that the environment comprising interconnected standalone applications. Participant P02.A suggested that Weki Input Helper should be integrated into Wekinator.

  b) Participants P01.A, P02.A and P04.A considered that MFEET's provided a limited level of agility and freedom to the configuration of the features for classification. Participant P04.A considered this was more critical during the exploration and selection of different feature sets for output and binding to Wekinator, which implied creating new projects with a different number of inputs.

4. *Comparison, evaluation of features, and general visualisation*

- Positive aspects:

  a) MFEET provided one way of comparing and evaluating the results of applying a specific feature extraction. This process used real-time, direct, subjective evaluation—e.g. the method of performing the different gestures or postures to generate data, and verifying that the classification and regression results and mapping of Wekinator were correct.

  b) The MFEET pipeline afforded a simple 'workaround' to support visualisation, by routing the outputs of the Weki Input Helper and Wekinator to Max, for instance.

- Opportunities for improvement:

  a) Participant P01.A and P02.A considered that MFEET provided limited means for exploration and feature comparison. There were suggestions about providing the functionality for running offline test beds to compare different features against data sets. There was no support showing the user measures of significance of a feature in training a model, for instance, which would allow the user to minimise the number of inputs (i.e. for improved learning of the model) or applying only a specific input.

b) Regarding visualisation, there was a strong consensus among participants that most of the interfaces of the pipeline required significant improvements. Participants P01.A, P02.A and P04.A suggested features for visualisation of the raw signal, of the processed signal after feature extraction, and better feedback about the classification and regression outcomes. Participants P01.A and P04.A found there was no way to provide visualisation of diagnosis of features or an assessment of their quality. When participant P04.A engaged in active exploration, he went back and forth to the source component—the sensor bridge, a Max object for capturing and visualising Myo data—which allowed him to visualise and analyse the signals obtained from different gestures. Participant P04.A found that the Myo sensor bridge, which I built principally as a component for sourcing data to the pipeline, was fundamental in the exploration of the signal and thanks to its visualisation features (e.g. real-time plotting, channel selection).

c) There were different parts in the pipeline that participants (P01.A P04.A) were considered obscure, such as the meaning of '#Outputs' in Wekinator and its relation to model building.

The results of the expert evaluation acknowledged MFEET as a strong proof-of-concept and yielded other valuable insights. I used these findings to inform the synthesis of the RAPID-MIX API design guidelines for feature extraction, which will be presented in section 5.4.3.

5.4.3 Discussion

I used the findings from the expert evaluation study with the MFEET prototype to draw insights and synthesise a set of design guidelines for the Feature Extraction (FE) subset of the RAPID-MIX API. These guidelines are contextualised and discussed below:

- *Provide feature extractors as modular and independent components* – modularity facilitates the adoption and integration of feature extractors in end-user applications and development environments. This is independent of whether it is in the context of high-level interfaces or text-based developer tools. Weki Input helper provides a useful set of feature extractors and signal processing components. This set of features can be provided off-the-shelf by a library of components with other features, such as i) bypass, ii) normalisation, iii) median, iv) envelope, v) modulus, vi) offset, vii) subtraction of the mean/average to compare signals, viii) filters (low pass, high pass and one-pole), ix) accumulator, x) RMS, xi) spectral feature extraction and xii) frequency domain convolution. Furthermore, providing high-level feature extractors as additional components, designed with domain knowledge to abstract the details of composing and parameterising low-level feature extractors with specific signals for specific applications—e.g. *Heart beat rate* for ECG, *Activation amplitude* and *Muscular activation* for EMG—can be very useful to support non-expert users. This will be further discussed in the next points.

- *Facilitate the integration and combination of FE and IML pipelines within the same environments* – having independent and modular components is desirable for sophisticated uses such as building bespoke signal processing pipelines for standalone applications, especially when using a text-based programming approach. For rapid prototyping of feature extraction pipelines, however, the lessons that high-level interfaces such as MFEET provide, is that operations—e.g. pre-selection, creation, composition, configuration, extension—should be integrated in the same environment. This supports users with an agile approach to the exploration of feature extractors with input signals and the assessment of their impact in the machine learning process, and in whole problem solving and exploration process.

  There are several insights that can be extracted for text-based programming approaches. For instance, providing users with scaffolding for end-to-end pipelines in the code, which include feature extraction with all stages, is paramount, especially for a learning stage. Supporting the extensibility of a FE library with architectural and design patterns (e.g. the strategy pattern for interchangeable segments of the pipeline such as feature extraction chains; a facade pattern for abstracting different components with similar functionalities). A desirable feature for both high-level interfaces and text-based developer tools which support rapid prototyping , exploration and structuring of the pipeline, is to support pipeline exports. This could be either a configuration in a serialisable format, code or a binary, that may integrate with target development environments or deploy in target applications.

- *Provide different abstraction levels to support both the novice user and the expert user* working with a FE and IML pipeline:

  a) In a high-level GUI-based interface, the novice-user track should support the user based on i) a wizard that makes *a strong proposition and constrains possibilities* and ii) provides pre-sets as building blocks. Pre-sets can be understood as high-level constructs grounded on real use cases which display domain-specific semantics. They may be abstracted from detail to be familiar to a target user group (*black-boxing*). These pre-sets wrap feature extractors or chains of feature extractors in association with a specific data source—for instance, “features for gesture recognition with an inertial motion unit sensor”, which encapsulate smoothing filters or an envelope follower. In an analogue way, for code-based developer tools, examples can be provided with full pipelines focusing on domain-specific scenarios, or segments of pipelines built with first principles, which can be abstracted into high-level components and made available for ready application to specific problems that fit novice end-user needs.

  b) The expert-user track should provide flexibility and a comprehensive collection of feature extractors, from the most basic filters, low-level feature extractors through to higher-level feature extractors. The expert user should also be provided with a high degree of flexibility in the structuring, manipulation, configuration and extension of feature extractors in chains.

- *Provide mechanisms for meaningful visualisations and feedback* that may assist users in the exploration of the different processing steps (from raw signal analysis of different channels, to post-processing stage, through to the visualisation of ML classification and regression outcomes). Examples using the RAPID-MIX API should provide these elements to support users' understanding. This could be achieved by providing a range of utility objects such as graph and plotting components, to complement learning in code examples or in extended unit tests fixtures (i.e. structural tests, with the full end-to-end pipelines and components).

- *Provide support for feature comparison and evaluation test beds in order to assist user decision.* For the user to assess the significance and efficacy of feature extractors in the IML process, they need to be compared against data sets and specific configurations of machine learning algorithms and parameters.

- *Provide mechanisms to maintain and persist the state of the FE and IML pipeline locally and remotely.* The metadata for the pipelines should be specified with an adequate depth, including chain, feature extractors, and feature extractor parameters. It is desirable to bundle different formats and cross-link with raw and processed data from the data and video repository to support pre-defined cases.

- *Provide for high-level interfaces to help users understand how to work with FE and IML.* High-level interfaces have shown to be fundamental for users (including developers) to understand IML. As such, background technologies such as Wekinator, Weki Input Helper or Gaussbox (i.e. a high-level interface built on top of XMM) should be included in the toolkit, to support pedagogical and hands-on exploration activities and potentially integrate with the RAPID-MIX API.

5.5 Synthesis of RAPID-MIX API design guidelines

In this section I attempt to illustrate the early stage of the reciprocal shaping between the RAPID-MIX API and the RAPID-MIX organisational context. I contextualise and illustrate the synthesis of RAPID-MIX API design guidelines using instances of stakeholders influencing design. The synthesis of the RAPID-MIX API design guidelines was informed by the goals of the consortium for the MIX products, the learning from findings that emerged from parallel prototyping and UCD actions, and different activities of the SIGs in the RAPID-MIX consortium. Activities in SIGs included discussions and the collaborative editing of *D2.3 Design guidelines for the RAPID-MIX API* (Mealla et al., 2016), for which I served as both a contributor and an internal reviewer.

In order to support a factual description of the synthesis in the following pages, I am including excerpts from *D2.3 Design guidelines for the RAPID-MIX API* (Mealla et al., 2016). I also assembled scattered pieces of primary data, which I have collected along the second action research cycle, mainly personal communications from different stakeholders and their discussions by email, which I put in context here.

5.5.1 General design guidelines

By the time of the writing of *D2.3 Design guidelines for the RAPID-MIX API* (Mealla et al., 2016), Mick Grierson (formally RAPID-MIX Innovation Manager) provided a high-level summary of the RAPID-MIX API directions:

> *The API is currently highly focused on the development of an approach for higher priority areas, where a solid API is not currently widely available, such as Machine Learning.*
>
> [...] *There is not a single, professional level API which can operate across all these platforms, and across a range of potential hardware including ARM processors, which includes sufficiently advanced features such as spectral feature extraction and frequency domain convolution. As these issues are a target priority for RAPID-*[MIX]*API, it may be useful to explore how best to address this.*
>
> [...] *There are few professional APIs that aim to simplify how to connect non-standard interaction and synthesis in meaningful, expressive ways. It is hoped that the RAPID-*[MIX] *API might do this well by providing simple sound APIs targeted at a range of simple, gestural 'instruments'.*

(Mealla et al., 2016, p. 28)

Here, Mick Grierson voiced the decision of the RAPID-MIX consortium to focus the RAPID-MIX API on machine learning, based on the recognition of an important technical gap. This gap consisted of the need for a cross-platform and multi-target API, designed with the 'right' set of advanced features, capable of accommodating a broad range of expertise. Also this API was specialised and oriented towards the implementation of real-time mappings between different domains, such as sensor-based interaction and control, and real-time audio. These would be the most distinguishing features of innovation of RAPID-MIX API as toolkit for rapid prototyping and product development.

Following the identification of the requirements for a cross-platform and multi-target API, the general design guidelines for the RAPID-MIX API recommended a multi-layer architecture comprised of a C++ API layer and a Javascript layer. General design guidelines also recommended a web platform that would host elements of the ecosystem surrounding the RAPIP-MIX API. These elements would include resources, such as documentation, code base, and an infrastructure of services for data storage and collaborative development environments. Therefore, besides the focus on machine learning and on a cross-platform and multi-target architecture, other significant aspects that were transversal to RAPID-MIX API design guidelines also prescribed for:

- support for user diversity – based on the RAPID-MIX user profiles that were refined in the UCD actions, and which comprised developers, designers, artists and academics—RAPID-MIX API was to support the needs of these kinds of users, and different levels of expertise in the continuum between novice and more advanced developers. Resources such as documentation and code examples were to be designed to take this into consideration and support exploration and rapid prototyping with ease. Core API modules should also provide for lower-level access and extensibility, for more advanced users.
- modularity and interoperability – SIGs recommended core ML, feature extraction, sensing devices and cloud-based data functionalities to be built as independent and compatible modules in RAPID-MIX API.
- easy integration of sensors – UCD actions showed that the lack of seamless integration between sensors at the source of machine learning pipelines, was an important deterrent and undermined the overall use of a machine learning pipeline. Providing easy integration of sensors, core machine learning modules, and services, and interoperability protocols such as OSC and Websockets, was an important RAPID-MIX API feature requirement.
- provision of an online ecosystem comprising resources, community, and infrastructure, including collaborative environments and data repositories – as reported by UCD actions, providing a centralised resource which unified the different RAPID-MIX API components, code repositories, and resources, such as documentation, tutorials, examples—including code and pre-existing data—and complementary building blocks. These elements were to be designed to enable self-starters, and to provide appropriate tracks for users of different expertise. Additionally, providing a community space that enables users to engage and interact with each other in a shared learning experience.

General RAPID-MIX API design guidelines also included generic API design heuristics, such as:

- *API documentation should be cohesive across the various implementation languages and API components*
- *API namespaces should be cohesive across the API components*
- *API should be highly modular*
- *API components should be licensed to include use in commercial products*

(Mealla et al., 2016)

Other areas of the RAPID-MIX API had specific guidelines defined. The following pages focus on the ML, feature extraction subsets of the RAPID-MIX API, and other API areas.

5.5.2 Machine Learning

By the time the initial design guidelines were documented in *D2.3 Design guidelines for the RAPID-MIX API* (Mealla et al., 2016) (July 31st, 2016), the focus of the RAPID-MIX API in IML was already acknowledged. The specific guidelines for the RAPID-MIX API ML subset included:

1. providing the right support for different levels of user expertise;
2. narrowing down the wide range of use cases;
3. reconciling the differences and overlap between IRCAM and GS approaches to IML;
4. dealing with the diversity of prototyping and deployment environments, and fulfilling the cross-platform and multi-target requirements for IML.

Eight months after the document's publication, in February 2017, Goldsmiths and IRCAM reached further consensus about the RAPID-MIX API ML subset. I introduce here an excerpt of Rebecca Fiebrink's personal communication to Goldsmiths team, which I formatted and highlighted some of the points in boldface for improved readability. This communication expresses some of the main points concerning the design guidelines for the RAPID-MIX API ML subset.

> *We believe that RAPID[-MIX] API for ML is most clearly distinguished from other ML APIs and Frameworks because of [its]:*
>
> *(1) need to support **fluid, easy interactions across multiple devices**,*
>
> *(2) need to support **streaming real-time data**,*
>
> *(3) need to support **end-user curation and modification of training examples**.*
>
> *In other words, it's about **supporting interaction between user & algorithm**, and **across multiple devices including mobile & Web, rather than about algorithms**. We may even want to provide hooks between our framework and existing API/frameworks like e.g. tensorflow. They are complementary.*

(R. Fiebrink, personal communication, Feb 2nd, 2017)

Most importantly, this communication illustrates the reinforced RAPID-MIX consortium decision to focus the RAPID-MIX API on Interactive Machine Learning, the approach to machine learning led independently both by Rebecca Fiebrink, at Goldsmiths, and Frederic Bevilacqua, at the SMMI group at IRCAM. This communication makes some of the uniqueness and innovativeness of the common approach to IML in RAPID-MIX more salient, particularly with respect to the type of interaction between user and ML algorithms, and the role of the end-user in curating and modifying training data.

Later on, and after the first release of the RAPID-MIX API, Michael Zbyszyński further contributed to the design guidelines with what he considered to be guidance of the implementation of the RAPID-MIX API ML subset. Here, I present a fragment of Zbyszyński's personal communication in an email to a group discussion thread about RAPID-MIX API core design ideas, where I highlighted the suggested API design guidelines using boldface:

> *What we need to do in this document is to draw some specific connections from our UCD research to the design of the API. For example:*
>
> - ***Get new users running as fast as possible****. We've found that users will not adopt a tool if they can't see it working in a few minutes.*
> - ***Provide many examples.*** *Projects like ofx, Processing and Arduino are successful because users can find examples that are similar to their needs. Also, the space of examples can demonstrate what a technology can do.*
> - *Where possible,* ***design functions based on use cases, not algorithms****. E.g.: a name like rapidmix::classification is better than one like rapidmix::svm.*
> - ***If we do use terminology, try to make it standard and worth knowing.*** *E.g.: classification and regression are technical terms, but it's worth the time for users to learn them because they are standard machine learning terms that would help them understand other machine learning.*
> - ***Minimise the number of required functions****. Most of our machine learning can run with three: 1) create, 2) train, and 3) run.*
> - ***Have sensible defaults*** *(This is connected to minimisation.) Users shouldn't have to figure out many parameters when they start. Functions should do something reasonable or typical if parameters are not set.*
> - ***Provide documentation that explains the concepts behind the API, not just the basic functions of the software***
>
> (M. Zbyszyński, personal communication, May 23rd, 2017)

Interestingly, we can observe how Zbyszyński, the RAPID-MIX API lead developer recognised the importance of UCD, when his was one of the critical voices of UCD in the different stages of the process. More importantly, this shows that he was able to reflect on the findings that emerged from the UCD process and distil guidelines from them. In Section 7.3, I will discuss the principles which were implicit in RAPID-MIX API design guidelines and which shaped the RAPID-MIX API.

5.5.3 Other RAPID-MIX API areas

Other RAPID-MIX API areas were assigned lower priority by the consortium and fall outside the scope of this thesis. To provide a brief overview, specific design guidelines for these areas in *D2.3*

Design guidelines for the RAPID-MIX API (Mealla et al., 2016) also included the challenges and overview of related work:

- Online repositories and collaborative communities – guidelines for this area of the API focused on the interoperability between IML and feature extraction components and the cloud-based services. Specifically, it prescribed formats and protocols—Javascript Object Notation (JSON) formats for online data storage and metadata, Representational State Transfer (REST) and real-time streams— for access, retrieval, and consumption by client applications, and operations for end-user data processing and visualisation.

- Sensors and hardware platforms – guidelines focused on interoperability and interconnectivity, and in their achievement through the definition of applicational and data protocols (i.e. on top of communication protocols such as OSC, Websockets) for devices, applications and services, bandwidth delimitation, provision of end-user programming capabilities for hardware, and performance and precision benchmarking across devices and use cases.

- Real-time audio synthesis and analysis – guidelines included the provision of an API design that was terse and easily parameterisable. Guidelines also included the provision of design features for use cases with immediacy requirements (rapid prototyping) and professional product development. Thus, the API was to be simultaneously useful to professional audio developers, and simple enough not to restrict the use by developers in the creative industries.

5.6 RAPID-MIX API: an ensemble and infrastructure emerging from RAPID-MIX

In this section, I proceed in the account and discussion of the RAPID-MIX API design process. I use the concept of ensemble IT artefact from Action Design Research (Sein et al., 2011), to discuss the reciprocal shaping and early development of the RAPID-MIX API within the technological and organisational context of RAPID-MIX up to its first version release. In the previous sections of this chapter, I provided an overview and discussion of the outcomes of parallel prototyping activities and the collaborative synthesis of the design guidelines RAPID-MIX API. This initial stage can be understood as part of the of *reciprocal shaping* (Sein et al., 2011) between the RAPID-MIX API as the design artefact and its designers within the organisational context of RAPID-MIX. In my overview of these activities, I also depicted instances of the *mutual influential roles* (Sein et al., 2011) in the design process, and of mutual learning between RAPID-MIX stakeholders. In this section, I proceed to examine the process of building the artefact, and how design guidelines and principles were translated into features.

The RAPID-MIX API intended to support rapid prototyping and product development by small and medium companies (SMEs), as well as by individual developers working in creative technology.

This included users with a diverse range of technical skills and expertise in software development, from professional audio developers, to designers, to artists, through to academics and computing students. Given the diversity in programming skills and lack of proficiency in machine learning of target user profiles, the RAPID-MIX API aimed to provide a low barrier to entry to machine learning, a smooth learning curve, and support for gradual progression through increasing complexity of use.

The RAPID-MIX consortium identified a variety of potential scenarios and requirements for a flexible API. The scenarios envisioned included use cases and potential products in the areas of musical technology, education, gaming, e-healthcare and sports. Use cases comprised applications integrating support for real-time sensor-based interaction, which would build upon the user context, for creative and expressive multimodal control and rich audiovisual output, and sonic augmentation of non-performative activities. Use cases also included heterogeneous development and deployment environments. Specifically, target development environments for using the RAPID-MIX API included creative development environments such as OpenFrameworks, JUCE, Max/MSP, Pd, Unity, and web frameworks. Target deployment environments for the RAPID-MIX API included desktop, mobile, web and embedded hardware applications.

There was also convergence towards focusing on supporting an interactive machine learning approach, which would focus on providing support for interaction between the end user and ML algorithms, end-user curation and modification of training examples, and a direct, intuitive, and subjective approach to the evaluation of ML model results. These requirements implied building the RAPID-MIX API as a toolkit that was robust and versatile enough to accommodate heterogeneous user needs, skills, goals and contexts of use. It also implied reconciling the gaps and overlap of the distinct technical approaches from the different RAPID-MIX consortium stakeholders.

5.6.1 A top-down perspective to RAPID-MIX API development

Early on in RAPID-MIX, consortium stakeholders envisioned the RAPID-MIX API as a unified piece of infrastructural software which was intended to fulfil the requirements summarised in the previous section. The RAPID-MIX API was designed to allow users to rapid prototype ML applications in multiple environments and devices, and to employ fast and easy-to-use workflows training, testing and using ML models. To meet the cross-platform and multi-target requirements (including desktop, browser-based, mobile, and embedded applications), the RAPID-MIX API was made available in both C++ and JS. The C++ API was intended for low-level audio and media developers, native mobile apps, and embedded processors. The JS API was intended for server-side (node.js) and client-side web applications, targeting both desktop and mobile browsers. Figure 5.5 depicts the general structure of the RAPID-MIX API as infrastructural software bearing a pipeline pattern and a modular architecture with core functional blocks containing different components. I will examine the development of these components in the next section.

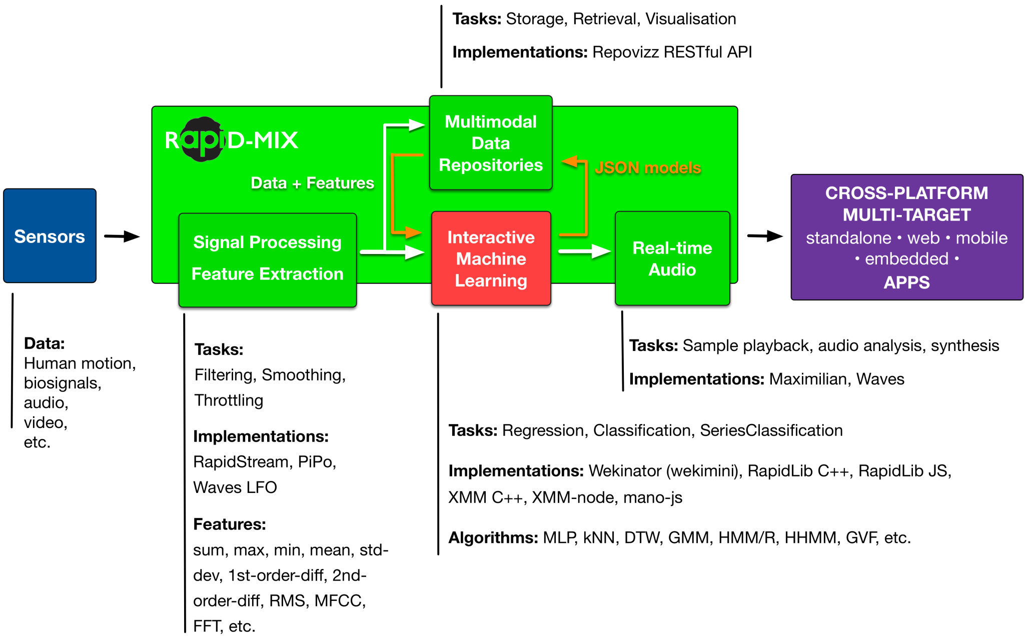

Tasks: Storage, Retrieval, Visualisation
Implementations: Repovizz RESTful API
RAPID-MIX
Data + Features
Multimodal Data Repositories
JSON models
Sensors
Signal Processing
Feature Extraction
Interactive Machine Learning
Real-time Audio
CROSS-PLATFORM MULTI-TARGET
standalone • web • mobile • embedded •
APPS
Data:
Human motion, biosignals, audio, video, etc.
Tasks:
Filtering, Smoothing, Throttling
Implementations:
RapidStream, PiPo, Waves LFO
Features:
sum, max, min, mean, std-dev, 1st-order-diff, 2nd-order-diff, RMS, MFCC, FFT, etc.
Tasks: Sample playback, audio analysis, synthesis
Implementations: Maximilian, Waves
Tasks: Regression, Classification, SeriesClassification
Implementations: Wekinator (wekimini), RapidLib C++, RapidLib JS, XMM C++, XMM-node, mano-js
Algorithms: MLP, kNN, DTW, GMM, HMM/R, HHMM, GVF, etc.

Figure 5.5: Diagram of the general structure of RAPID-MIX API

The RAPID-MIX API implemented the overarching concept of the pipeline and provided the unifying "glue" for some of the components previously depicted in Figure 5.5. RAPID-MIX API implemented a single top-level interface intended to simplify and abstract the common, complex functionalities of the underlying component modules to the stages of a machine learning pipeline. The top-level interface aggregated and provided simplified adapters to the underlying modules. RAPID-MIX API lead developer Zbyszyński designed and implemented the RAPID-MIX API top-level interface. Larralde and I contributed with the integration of some of the ML underlying modules—XMM and GVF respectively—and the implementation of their adapter classes. Specifically, the top-level interface provided:

a) a common interface to underlying modules which implemented supervised ML algorithms, and that were selected for sharing the ability to create expressive models from small training data sets, and low-training and run times (Fiebrink et al., 2011). These modules included: 1) RapidLib (GS), which implemented k-nearest neighbour and the multilayer perceptron algorithms (Figure G.1), 2) XMM (Françoise et al., 2013) (IRCAM), which implemented Gaussian Mixture Models or Hidden Markov Models (Figure G.2), and 3) Gesture Variation Follower (Caramiaux et al., 2014) (GS/IRCAM), which implements particle filtering.

b) a common interface to different underlying modules that implemented signal processing and feature extraction primitives—e.g. Maximilian (GS) which implements, for instance low pass filters, Root Mean Square, Mel frequency cepstral coefficients or Fast Fourier Transform; or, other modules for signal processing and feature extraction primitives that provided varying trade-offs between simplicity, sophistication of use and power, such RapidStream (GS) (simpler) and PiPo (IRCAM) (more flexible and powerful but also more complicated).

c) an interface for a generic data structure to allow users to explore different ML classes with the same data set. The interface provided functionalities for managing data and data sets and was compatible with all of the ML classes. This generic data structure provided a vector for input values, representing sensor or media features for use in both the training and test ML stages, and a vector for output values, for the results of application of the ML model in the test stage to new data. Additionally, this interface provided functionalities to facilitate interoperability between modules, exposing JSON serialisation and deserialisation functions;

Listing 5.1: RAPID-MIX API “Hello World” example in C++

```cpp
#include <iostream>
#include "rapidmix.h"

int main(int argc, const char * argv[]) {
    //Create a machine learning object for regression
    rapidmix::staticRegression mtofRegression;

    //Create an object to hold training data
    rapidmix::trainingData myData;

    //Set up the first element of training data
    std::vector<double> input = { 48 };
    std::vector<double> output = { 130.81 };
    myData.addElement(input, output);

    //Add more elements
    input = { 54 };
    output = { 185.00 };
    myData.addElement(input, output);

    //Train the machine learning model with the data
    mtofRegression.train(myData);

    //Get some input
    int newNote = 0;
    std::cout << "Type a MIDI note number.\n";
    std::cin >> newNote;

    //Run the trained model on new input
    std::vector<double> inputVec = { double(newNote) };
    double freqHz = mtofRegression.run(inputVec)[0];

    std::cout << "MIDI note " << newNote;
    std::cout << " is " << freqHz << " Hertz" << std::endl;
}
```

The implementation of the RAPID-MIX API top-level interface also intended to minimise the number of steps and simplify the interface of the objects required to create a functional prototype. A vital design assumption in the RAPID-MIX API was that, rather than focus on the details of specific ML algorithms, a user building a new system should be encouraged to focus on their design goals. Specifically, the RAPID-MIX API encourages users to focus on setting up a minimal pipeline, the expected inputs and desired outputs, and assessing whether the trained model performs satisfactorily according to users' criteria for the design task. As the RAPID-MIX API "Hello World" example shows (Listing 6.1), users need only to set up an instance of an ML class, populate a data set which determines the dimensionality of the ML model inputs, train an ML model on the data set, and run the model on new data.

The RAPID-MIX API implementation embodied a certain amount of ML expertise through algorithm curation, default parameter setting, and black-boxing through concept mapping. Figure 5.6b depicts an early sketch of the RAPID-MIX API conceptual model, which organised ML modules in an interface with two design tasks and two types of inputs. The two design tasks correspond to supervised ML tasks, classification or regression, which entail the assignment of outputs from the ML models, respectively, labels as discrete categories or continuous numerical values. The types of inputs refer to the temporal nature of data—i.e. static or temporal data, corresponding to individual instances or time series (for example, a hand pose or a hand gesture over time). The conceptual model of XMM—which similarly structured different Hidden Markov model algorithms (Figure a) in a two-by-two matrix—inspired the conceptual model of the RAPID-MIX API ML interface. The core RAPID-MIX API ML classes reflect this structure, for example, "rapidmix::classification", "rapidmix::xmmTemporalClassification" or "rapidmix::gvfTemporalVariation'. Default ML algorithm parameter values were designer-defined with regards to a set of common use cases.

There were other essential assumptions in the design of the RAPID-MIX API which are worthy of note. One of these assumptions was that users understood the pipeline architecture, which is common to many real-time applications in audio and sensor-based systems. Another assumption was that users understood basic and standard ML terminology such as classification or regression. Both of these assumptions were covered by the RAPID-MIX API documentation, which provided an introduction to these concepts (e.g. standard machine learning terminology) and their application in design to multiple application examples. The RAPID-MIX API website[2] centralised access to the different documentation resources, such as the general introduction to ML concepts, getting started, tutorials, reference, links to the codebase, and very importantly, an extensive set of examples.

One final important design assumption concerned ML model evaluation and originated from the lessons learned through Wekinator (Fiebrink et al., 2011). When training data is provided interactively, as in the main workflow encouraged by RAPID-MIX API, direct observation of a model's behaviour on new data is often the most effective way to assess a model's performance (and evalu-

[2] The RAPID-MIX API official website, http://www.rapidmixapi.com

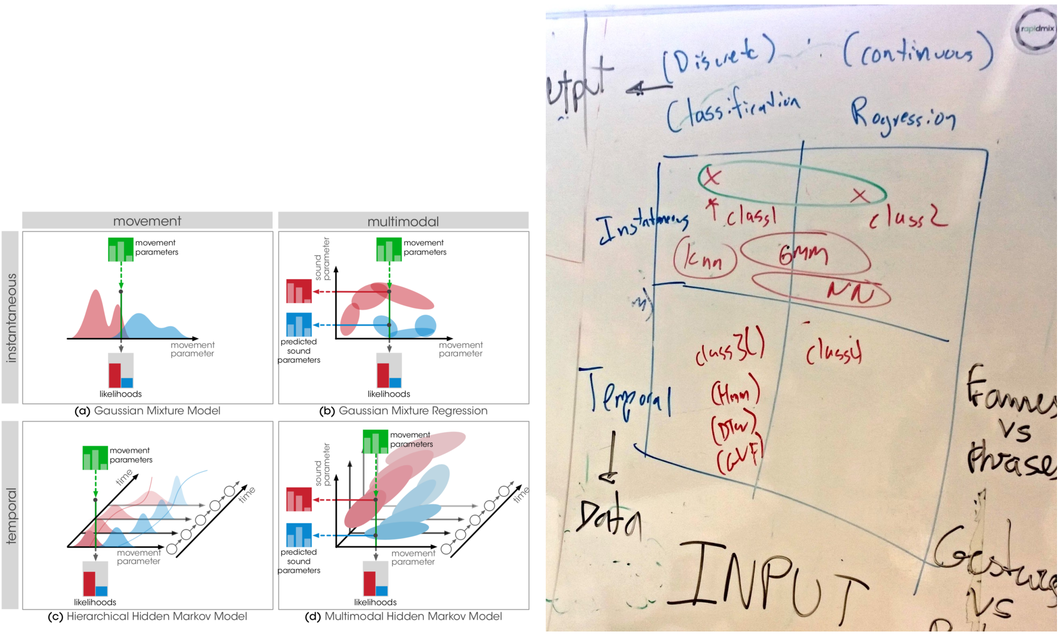


Figure 5.6: a) Schema for the conceptual model of probabilistic ML models in XMM (Françoise et al., 2014), according to the temporal nature of data and type of information, and b) early sketch of the conceptual model of RAPID-MIX API ML interfaces in the Goldsmiths EAVI lab whiteboard

ation using more conventional metrics such as cross-validation can be misleading) (Fiebrink et al., 2011). As such, given the nature of IML workflow and the end-user goals the RAPID-MIX API is intended to provide for—i.e. to develop prototypes and 'throw-away' ML models, to directly observe the results of the application of the trained model to new data in real-time, and subjectively evaluate the model's performance—it does not provide features such as cross-validation or training accuracy of the ML models.

5.6.2 A bottom-up perspective to RAPID-MIX API development

From early development until the release of the first version of RAPID-MIX API, different RAPID-MIX consortium stakeholders contributed to the production of an ensemble of components that integrated different RAPID-MIX API functional areas. Primarily, these components were software libraries—tools and building blocks that targeted developers with varying levels of technical skills and proficiency of the different domains of interest—and which could independently integrate different types of software projects, including different languages, target platforms and architectures. These libraries, which Table 5.4 characterises, concern users, use case and type of prototypes, target platform and authors, and are briefly described as follows:

- RapidLib C++ (GS) – C++ ML API for regression and classification.
- RapidLib.js (GS) – Javascript ML API for regression and classification, transpiled from RapidLib C++ using emscripten[3].

[3] A process and software framework for engineering cross-platform libraries by porting C++ to asm.js via *em-*

- MaxiLib.js (GS) – (prototype #9) Javascript audio synthesis and signal processing library transpiled from *Maximilian* using *emscripten*
- xmm-node (IRCAM) – a Node.js module (server-side executing Javascript) wrapping the XMM library for statistical ML modelling with Hierarchical Hidden Markov Models.
- xmm-client (IRCAM) – a client-side browser Javascript library for structuring and managing datasets and XMM model decoders.
- waves-lfo (IRCAM) – graph-based Javascript API with components—Low Frequency Operators (LFO)
  —designed for the signal processing of real-time event data streams such as audio, audio descriptors and motion sensor data.
- xmm-lfo (IRCAM) – Javascript low frequency operator wrappers of xmm-client classes for waves-lfo.
- lfo-motion (IRCAM) – Javascript modules targeted for movement analysis.
- Mano.js (IRCAM) – Javascript library that provides a high-level client-side wrapper of waves-lfo, lfo-motion, xmm-client, for sensor processing, and gesture modeling and recognition.
- PiPo Plugin in JUCE (IRCAM) – JUCE plugin for dynamic loading of PiPo chains and operators for real-time signal processing.
- JUCE ML Module (JUCE) – JUCE plugin wrapping RapidLib C++ ML models for regression and classification.
- RepoVizz 2.0 API (MTG/UPF) – a RESTful API client (C++/JS) for Repovizz 2, the next-generation cloud-based service for multimodal data storage.

Some of the technological bricks in Table 5.4 resulted from integrations of background technologies with platforms or new engineering processes—for instance, *MaxiLib.js* and the *PiPo plugin in JUCE* (prototype #10), consisted of the PiPo codebase encapsulated as a JUCE plugin and made available for audio software developers who use the JUCE platform. Others technological bricks such as Repovizz 2 API were implemented from scratch.

Except for Repovizz 2 API, which was developed by MTG/UPF, Goldsmiths and IRCAM produced most of the technological bricks in Table 5.4 that were related to the signal processing, feature extraction and machine learning functionalities, and which became the C++ and Javascript libraries that comprised the RAPID-MIX API. As previously presented, both IRCAM and Goldsmiths led independent research in Interactive Machine Learning (Fiebrink et al., 2011; Françoise et al., 2013). The decision to fixate the RAPID-MIX API design direction on an IML approach could appear evident at the beginning. However, it implied identifying and reconciling the gaps and overlaps of the technical approaches of both RAPID-MIX consortium institutions.

scripten (http://emscripten.org)

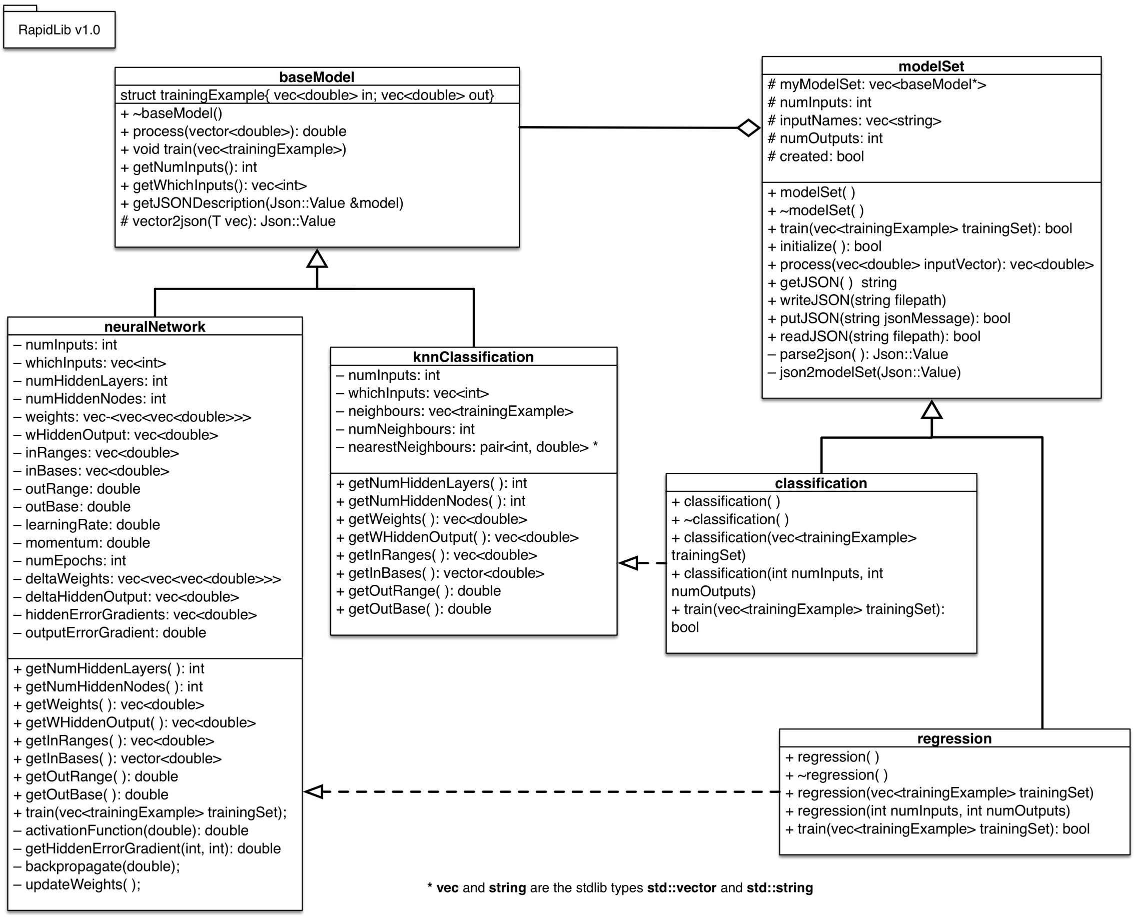

RapidLib v1.0
baseModel
struct trainingExample{ vec<double> in; vec<double> out}
+ ~baseModel()
+ process(vector<double>): double
+ void train(vec<trainingExample>)
+ getNumInputs(): int
+ getWhichInputs(): vec<int>
+ getJSONDescription(Json::Value &model)
# vector2json(T vec): Json::Value
modelSet
# myModelSet: vec<baseModel*>
# numInputs: int
# inputNames: vec<string>
# numOutputs: int
# created: bool
+ modelSet( )
+ ~modelSet( )
+ train(vec<trainingExample> trainingSet): bool
+ initialize( ): bool
+ process(vec<double> inputVector): vec<double>
+ getJSON( ) string
+ writeJSON(string filepath)
+ putJSON(string jsonMessage): bool
+ readJSON(string filepath): bool
– parse2json( ): Json::Value
– json2modelSet(Json::Value)
neuralNetwork
– numInputs: int
– whichInputs: vec<int>
– numHiddenLayers: int
– numHiddenNodes: int
– weights: vec-<vec<vec<double>>>
– wHiddenOutput: vec<double>
– inRanges: vec<double>
– inBases: vec<double>
– outRange: double
– outBase: double
– learningRate: double
– momentum: double
– numEpochs: int
– deltaWeights: vec<vec<vec<double>>>
– deltaHiddenOutput: vec<double>
– hiddenErrorGradients: vec<double>
– outputErrorGradient: double
+ getNumHiddenLayers( ): int
+ getNumHiddenNodes( ): int
+ getWeights( ): vec<double>
+ getWHiddenOutput( ): vec<double>
+ getInRanges( ): vec<double>
+ getInBases( ): vector<double>
+ getOutRange( ): double
+ getOutBase( ): double
+ train(vec<trainingExample> trainingSet);
– activationFunction(double): double
– getHiddenErrorGradient(int, int): double
– backpropagate(double);
– updateWeights( );
knnClassification
– numInputs: int
– whichInputs: vec<int>
– neighbours: vec<trainingExample>
– numNeighbours: int
– nearestNeighbours: pair<int, double> *
+ getNumHiddenLayers( ): int
+ getNumHiddenNodes( ): int
+ getWeights( ): vec<double>
+ getWHiddenOutput( ): vec<double>
+ getInRanges( ): vec<double>
+ getInBases( ): vector<double>
+ getOutRange( ): double
+ getOutBase( ): double
classification
+ classification( )
+ ~classification( )
+ classification(vec<trainingExample> trainingSet)
+ classification(int numInputs, int numOutputs)
+ train(vec<trainingExample> trainingSet): bool
regression
+ regression( )
+ ~regression( )
+ regression(vec<trainingExample> trainingSet)
+ regression(int numInputs, int numOutputs)
+ train(vec<trainingExample> trainingSet): bool
* vec and string are the stdlib types std::vector and std::string

Figure 5.7: RapidLib v1.0 class diagram

On Goldsmiths' side, Wekinator (Fiebrink et al., 2011) provided an intuitive and high-level GUI-based toolkit for IML. Despite Wekinator's support for the integration of machine learning models with other applications via the OSC communication protocol in a desktop environment, it did not provide a way to integrate ML models' code into other applications. This limitation precluded important applications in product development scenarios, such as the robust integration of native ML models and workflow implementations in mobile and embedded devices, and web applications. This limitation was addressed with Wekinator model exporter[4] (Table 5.4), a class that Zbyszyński developed as he arrived in RAPID-MIX, and that extended Wekinator with functionality for the generation of C++ source code of Wekinator-trained ML models and their persistence to the file system, for subsequent integration in client application codebases.

RapidLib originated from Wekinator model exporter. The predominant adopted approach "was the simplest" and the most minimal—starting with the generation via Wekinator of hard-coded C++ trained ML models with "no training features or exposed parameters" and extracting these models into an independent C++ library—Appendix G shows the inception of RapidLib through the history of commits to Wekinator (Section G.1, Appendix G) and its design rationale in a personal communication by Zbyszyński (Section G.2, Appendix G).

[4] https://github.com/mzed/wekimini

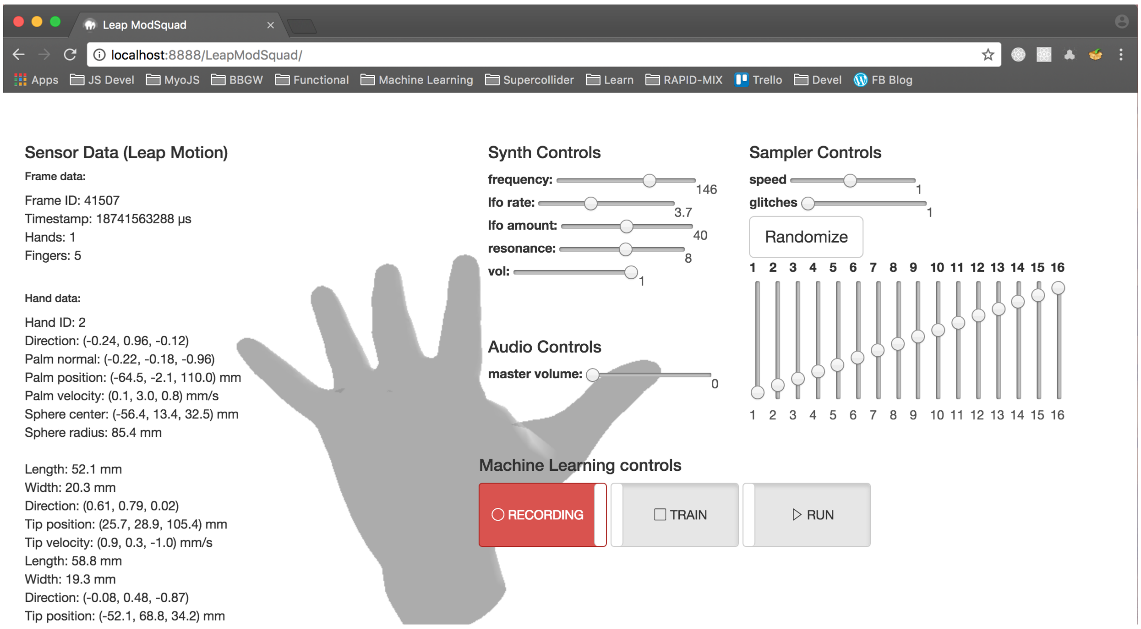
As the UML class diagram shows (Figure G.1), RapidLib v1.0 had very simple architecture with no public parameters. RapidLib v1.0 implementation abstracted the k-nearest neighbour algorithm and the multi-layer perceptron into a *classification* and *regression* classes—i.e. the design tasks. This is also congruent with the RapidLib v1.0 documentation (Section G.3, Appendix G), which stated that default parameters for the algorithms were hard-coded. Further development throughout the RapidLib design process expanded and increased its complexity, with the addition of algorithms and more parameters exposed to the end user. RapidLib was also transpiled into a JS library using Emscripten (Zakai, 2011), based on the learning provided by Maximilian and MaxiLib.js.

On IRCAM side, XMM (Françoise et al., 2013) was a powerful library focusing on temporal models for gesture recognition. This was a very desirable and complementary feature to the overall set of RAPID-MIX API ML functionalities, given that, at the beginning of RAPID-MIX, Wekinator lacked support for temporal models. This gap was only addressed with Wekinator 2.0 featuring DTW, by the end of the RAPID-MIX first year, and later on, after the first release of RapidLib. As shown by Larralde's XMM UML class diagram (Section G.2, Appendix G), XMM had a rather complex implementation and API, when compared to RapidLib. XMM had a significantly wider surface area, higher number of classes and exposed parameters, more complex data structures for configuration, input and output for ML models, and an event system which required subscription to ML model results. XMM provided more robust temporal models for gesture recognition but also more expertise in configuration. It also required more processing power for the training process, and executed natively on a device or a server-side installation. Throughout the design process, XMM was simplified, through the extraction of more abstract layers and wrappers, such as XMM-node and Mano.js.

These distinct technologies and approaches developed differently but in a convergent fashion throughout the RAPID-MIX API design process. RapidLib v1.0 was first released internally within the RAPID-MIX consortium and then evaluated with end-users in the wild, in the first UCD actions for formative evaluation that interwove the building of the RAPID-MIX API. XMM wrappers were released around the same time and made available also in the UCD actions for evaluation of the RAPID-MIX API. These UCD actions for formative evaluation of RAPID-MIX API will be examined in Section 5.7.

5.6.3 My software development contributions to RAPID-MIX API

My software development contributions to RAPID-MIX API focused on providing useful functionality and user resources to inspire the adoption and familiarisation with the RAPID-MIX API. Additionally, these contributions aimed to support the subsequent evaluation of the toolkit. Specifically, they included online, real-time, end-to-end IML pipelines build with RapidLib.JS for CodeCircle and independent web applications. These applications were *Core Test Applications* (Edwards et al., 2003), which contributed to evaluate the core features of the RAPID-MIX API



Sensor Data (Leap Motion)

Frame data:

Frame ID: 41507
Timestamp: 18741563288 µs
Hands: 1
Fingers: 5

Hand data:

Hand ID: 2
Direction: (-0.24, 0.96, -0.12)
Palm normal: (-0.22, -0.18, -0.96)
Palm position: (-64.5, -2.1, 110.0) mm
Palm velocity: (0.1, 3.0, 0.8) mm/s
Sphere center: (-56.4, 13.4, 32.5) mm
Sphere radius: 85.4 mm

Length: 52.1 mm
Width: 20.3 mm
Direction: (0.61, 0.79, 0.02)
Tip position: (25.7, 28.9, 105.4) mm
Tip velocity: (0.9, 0.3, -1.0) mm/s
Length: 58.8 mm
Width: 19.3 mm
Direction: (-0.08, 0.48, -0.87)
Tip position: (-52.1, 68.8, 34.2) mm

Synth Controls

frequency: 146
lfo rate: 3.7
lfo amount: 40
resonance: 8
vol: 1

Sampler Controls

speed 1
glitches 1

Audio Controls

master volume: 0

Machine Learning controls

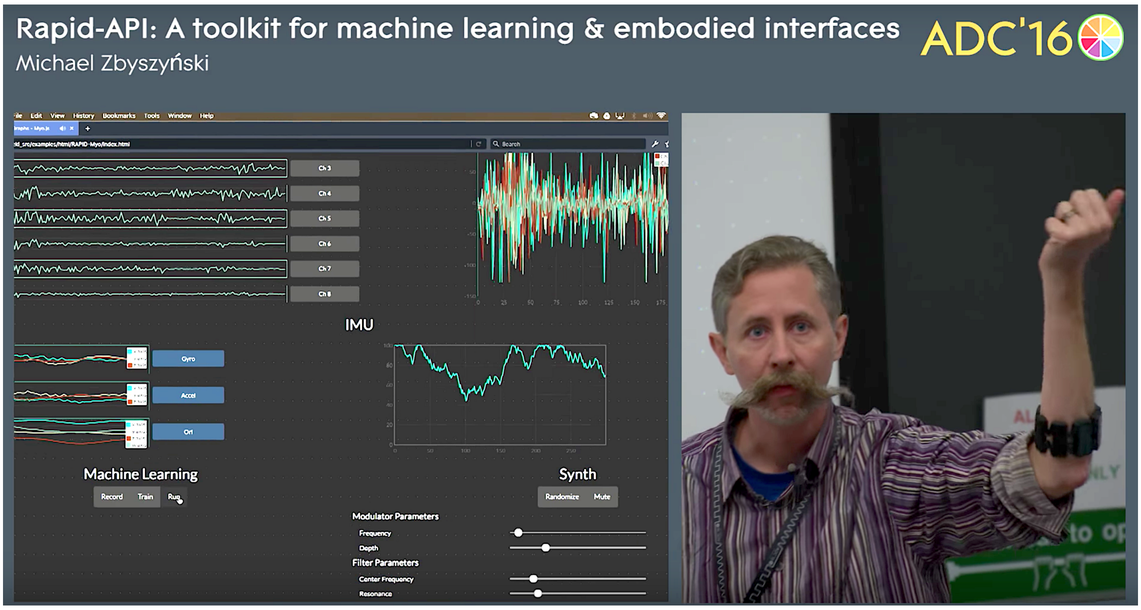
Figure 5.8: RAPID-LeapModSquad, a demonstrator for RapidLib.js and Maxim.js, which I built in collaboration with Adam Parkinson.

and eventually become RAPID-MIX API demonstrators. Other contributions included the integration of GVF in RAPID-MIX API ML interface and introduction of a unit-test fixture to test and demonstrate the use of RAPID-MIX API C++ layer.

From the possible alternatives for building demonstrators for the RAPID-MIX API, I chose the Web, and in particular, CodeCircle as the medium for its development and deployment. CodeCircle (Fiala et al., 2016) is a Web-based collaborative online coding environment for JavaScript, HTML and CSS, that presents many desirable features for teaching of creative computing. For instance, it supports the main development tasks in a single environment, such as live coding and execution, assets upload, provides a default set of multimedia libraries—including RapidLib.js library into CodeCircle (Zbyszyński et al., 2017)—and allows the easy deployment and integration of 3rd-party libraries, facilitates collaborative editing and code sharing.

I designed and implemented different prototypes as RAPID-MIX API core test applications and demonstrators using the RapidLib.js subset and CodeCircle documents. The design rationale for my RAPID-MIX API demonstrators included:

- To exemplify how to integrate IML workflows with live data streams from different off-the-shelf, consumer-based multiparametric sensors (e.g. Myo, LeapMotion) for the control of real-time audio processes, using RapidLib.js and client-side vanilla Javascript.
- To encourage the adoption of and experimentation with RapidLib.js by self-learners and ML novice end-users by providing them with building blocks to employ in their applications and designs.
- To build upon the affordances of the World Wide Web for rich internet applications, such as



Rapid-API: A toolkit for machine learning & embodied interfaces

Michael Zbyszyński

ADC'16

Figure 5.9: M. Zbyszyński presenting my RAPID-MIX API demonstrator *RAPID-MYO* at ADC'16

high availability, high distributability, ease of deployment with "zero install", and intuitive, user-friendly, visual metaphors in the user interface.

I designed and implemented two prototypes, *LeapModSquad* and *RAPID-Myo*, as functional RapidLib demonstrators for applying regression as a design task. In these prototypes, regression was applied to build mappings between multi-parametric sensors (e.g. Myo, LeapMotion) and real-time multi-parametric audio processes, for musical control and expression. For instance, in LeapModSquad (Figure 5.8), the Leap Motion sensor generated values (inputs to RapidLib regression) for a specific hand position sensed within its range. These values were associated with a set of parameters of two audio processes built with Maxim.js (outputs and labels)—a sampler (which Adam Parkinson contributed for the implementation) and a synthesizer.

In *RAPID-Myo*[5] (Figure 5.9), the Myo armband inertial motion unit sensor generated 10 values for 3D orientation and motion of the arm, and 8 values from EMG sensors for the muscle tension of the forearm. These values provided the features and inputs to the IML workflow. With *RAPID-Myo*, I also implemented other functionalities to demonstrate the interactive machine learning workflow such as feature selection. For instance, in RAPID-Myo, the user can select the specific channels of data values and compose a variable-sized frame for inputs to RapidLib regression—e.g. orientation-x-axis, or composing 4 channels of EMG channels with the accelerometer x axis. Input values were associated with a set of parameters of a basic synthesizer (outputs), implemented with Maximilian JS.

I developed another RapibLib JS demonstrator to apply classification as a design task for customising gestural control for the Spotify player with Leap Motion. This application was developed "in the wild", at the 24-hour ANVIL Hack III. Figure 5.10 shows the GUI of RAPID-Leapify.

[5] This presentation is available on JUCE's youtube channel, `https://youtu.be/8IEVWj_0YhM?t=1414`

I built this ‘hack’ at the 24-hour ANVIL Hack III, where it was presented, having received the Spotify award. RAPID-Leapify enables the end user to record custom gestures using LeapMotion and a train model that maps the gestures to Spotify Player Web API operations—i.e. *Play*, *Pause*, *Skip to next* and *Previous track*. When in running mode, RAPID-Leapify triggers these commands upon enactment of the hand gestures within LeapMotion range.

These RAPID-MIX API demonstrators were utilised in different ways. For instance, my colleague Zbyszyński included these demonstrators on the RAPID-MIX API website in the gallery for RAPID-MIX API community examples. They were also used in promotional and dissemination activities of the RAPID-MIX project—e.g. Zbyszyński also demonstrated one of my prototypes at the Audio Developers Conference 2016 organised by JUCE/ROLI (Figure 5.9). Most importantly, they were used as research materials for two UCD actions, which included a 2-week workshop with creative coders and a 1-day workshop at the AudioMostly conference. Participants in these actions opted to use these demonstrators as building blocks for their own projects, as I will examine in Section 5.7.3.

5.7 User-Centred Design Actions for formative evaluation of RAPID-MIX API

In the next sections, I describe two selected UCD actions which investigated the usability and appropriation of two different subsets of the RAPID-MIX API by different types of users. These UCD actions took place in months 23 and 29 of the RAPID-MIX project.

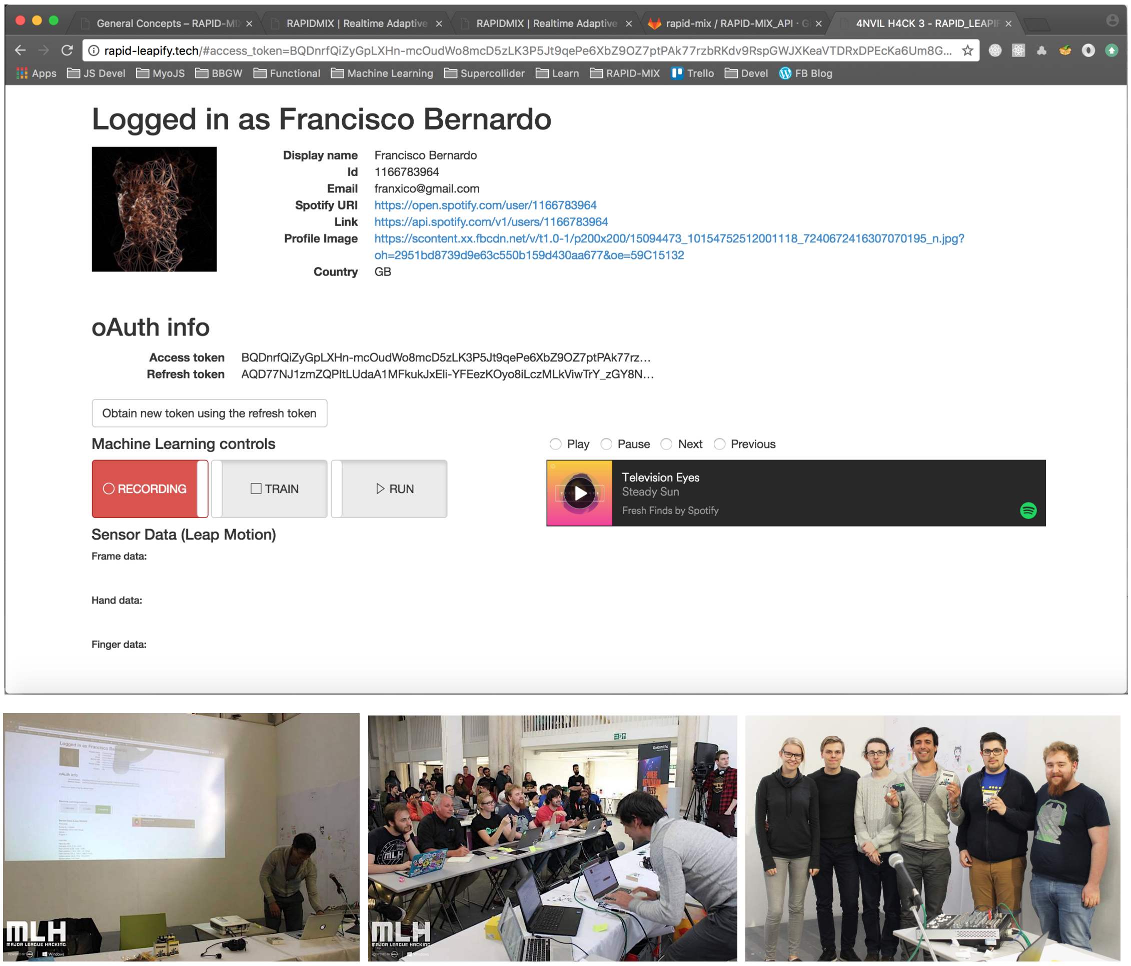
5.7.1 JUCE Machine Learning hackathon with audio developers

This section describes a UCD action that investigated the usability and appropriation of a specific subset of the RAPID-MIX API, the RapidLib C++ with professional audio software developers. In this UCD actio, which took place in month 23 of the RAPID-MIX project, I wanted to investigate the following questions:

- What are the needs, goals and values of this user group?
- How do these users use the RAPID-MIX API and what for? What aspects of machine learning or of the RAPID-MIX API was confusing for these users? What errors did they make? What was unexpected?
- What other API features might they need, and which would they need the most?

The JUCE Machine Learning Hackathon (Figure 5.11) was a one-day hackathon in December 2016, organised with ROLI[6], an SME in music technology that participates in the RAPID-MIX

[6] ROLI, https://roli.com


General Concepts – RAPID-MI
RAPIDMIX | Realtime Adaptive
RAPIDMIX | Realtime Adaptive
rapid-mix / RAPID-MIX_API · G
4NVIL H4CK 3 - RAPID_LEAPIF
Apps
JS Devel
MyoJS
BBGW
Functional
Machine Learning
Supercollider
Learn
RAPID-MIX
Trello
Devel
FB Blog
MLH
MLH

rapid-leapify.tech/#access_token=BQDnrfQiZyGpLXHn-mcOudWo8mcD5zLK3P5Jt9qePe6XbZ9OZ7ptPAk77rzbRKdv9RspGWJXKeaVTDRxDPEcKa6Um8G...

Logged in as Francisco Bernardo

Display name	Francisco Bernardo
Id	1166783964
Email	franxico@gmail.com
Spotify URI	https://open.spotify.com/user/1166783964
Link	https://api.spotify.com/v1/users/1166783964
Profile Image	https://scontent.xx.fbcdn.net/v/t1.0-1/p200x200/15094473_10154752512001118_7240672416307070195_n.jpg?oh=2951bd8739d9e63c550b159d430aa677&oe=59C15132
Country	GB

oAuth info

Access token	BQDnrfQiZyGpLXHn-mcOudWo8mcD5zLK3P5Jt9qePe6XbZ9OZ7ptPAk77rz...
Refresh token	AQD77NJ1zmZQPltLUdaA1MFkukJxEli-YFEezKOyo8iLczMLkViwTrY_zGY8N...

Obtain new token using the refresh token

Machine Learning controls

○ RECORDING □ TRAIN ▷ RUN

○ Play ○ Pause ○ Next ○ Previous

Television Eyes
Steady Sun
Fresh Finds by Spotify

Sensor Data (Leap Motion)

Frame data:

Hand data:

Finger data:

Figure 5.10: a) The GUI of RAPID-Leapify, an online web application that features the IML workflow for training custom hand-gestures using the Leap Motion to control the Spotify Player Web API. I built this ‘hack’ at the 24-hour ANVIL Hack III b) c) where it was presented d) receiving the Spotify award.

consortium. Part of ROLI’s product portfolio, JUCE[7] is a popular cross-platform C++ framework with a focus on audio applications, which is widely used in the industry. JUCE’s customers and users are audio software engineers, developing audio and music apps for different platforms (standalone applications and plug-ins for desktop, and mobile apps for iOS and Android). The hackathon focused on the JUCE RAPID-MIX Module, a thin wrapper around the RAPID-MIX API which exposed functions for training and evaluating classification and regression models, utility functions for model serialization/de-serialization from JSON (JavaScript Object Notation) files, and data structures with JUCE primitive data types.

The JUCE Machine Learning Hackathon was advertised on an online booking site, on the JUCE forum and mailing lists, and mailing lists of educational institutions. To incentivise participants, a ROLI Lightpad BLOCK [3] was given as an award for each of the winning team’s attendees. The hackathon began with an induction introducing participants to supervised machine learning techniques and to the JUCE RAPID-MIX Module. Consent forms were distributed along with

[7] JUCE, https://juce.com

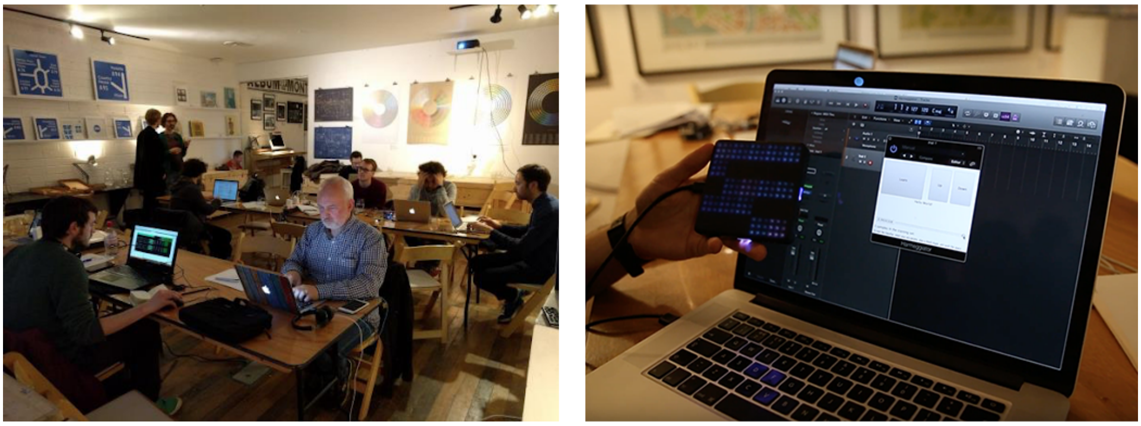


Figure 5.11: a) The JUCE Machine Learning Hackathon and b) the winning 'hack' *Harmeggiator*

a short pre-hack questionnaire about participants' skills in software development, programming languages and environments, and machine learning.

Participants then had six hours to complete a "hack"—a small project of their choosing that used the JUCE RAPID-MIX Module—after which every team presented their hack to a jury panel of JUCE and RAPID-MIX representatives. Participants worked in groups of no more than three. Teams posted their hack code to GitHub. Hackathon facilitators recorded questions, critiques and feedback that were voiced by the participants. After the awards ceremony, we conducted structured interviews with participants. We interviewed them about changes in their design goals throughout the hack, module features that they used, limitations they discovered and strategies they used to overcome them, and suggestions for real-world applications of the module. The interviews were video recorded and subsequently analysed.

5.7.2 Results

20 developers enrolled in the hackathon. 12 developers actually attended the event, including 3 ROLI developers. The pre-hack questionnaire indicated that most of the participants were proficient in C++, had extensive programming knowledge, and had used JUCE before. Most participants stated they had very limited or no knowledge of machine learning techniques. Most teams had a clear idea of what they wanted to build for their hack after the ML presentation.

Some participants indicated that they did not intend to submit a hack. Rather, they participated to learn about machine learning. Five teams submitted a hack. We briefly describe each hack as follows:

- P01.B – *Embedded ML* – This hack ran the JUCE RAPID-MIX module on Beagle Bone Black with Bela (McPherson, 2017), a highly constrained embedded system tailored for ultra-low-latency audio. This system used ML for gesture recognition with a ROLI Lightpad BLOCK. The author interestingly stripped the JUCE wrapper code away and used the RAPID- MIX API directly in a console application.

- P02.B – *Filter Classification* – used ML as a quick prototyping alternative for digital audio filter design. This system classified filter types (i.e., high-pass, low-pass, band-pass) from a set of coefficients of a Finite Impulse Response (FIR) filter—i.e., feedforward filter type with a finite duration impulse response (Steiglitz, 1996).

- P03.B – *Harmeggiator* – implemented a MIDI effect VST plugin (Virtual Studio Technology by Steinberg [11]) to arpeggiate chords from gestures performed with the ROLI Lightpad BLOCK. The system provided functionalities for applying the IML workflow to train and map arpeggiation parameters (speed, arpeggiation direction, shape) extracted from gestures to a set of intervals extracted from chord note values (Figure 2b).

- P04.B – *Feature Extractor + RapidMix* – extended this participant's existing audio feature extraction software with IML capabilities. IML was applied to the audio analysis features and used to drive generic parameters of an audiovisual application.

- P05.B – *FM Synth Patch Generator* – calculated FM synthesis (Chowning, 1973) parameters to match synthesizer output with sampled instruments through similarity analysis of audio—i.e., to make an FM synthesizer resemble the sound of a recorded piano. Technical and conceptual issues prevented the timely delivery of a fully functional hack.

In general, observation and feedback from interviews about the use of JUCE RAPID-MIX module confirmed it was an appropriate tool to achieve the users' proposed hacks and prototypes. There was highly positive feedback about module code quality and clarity, and the fast implementation results it enabled. Documentation and examples were found easy to navigate and use. The diversity in the type of applications submitted showed that the JUCE RAPID-MIX module is useful and usable for a broad range of applications and for a variety of hardware platforms. One participant mentioned that introductory talks delimited the state and capabilities of the library very well and that this influenced the scope of what the participant wanted to do.

There were usability issues identified along with other technical issues, including:

- Participant P03.B was confused by one of the higher-level abstractions built into the API that targeted a use case that was not relevant to these hacks (specifically, an abstraction aggregating many ML models into one data structure for use in multiparametric synthesiser mapping).

- Participants P01.B, P03.B and P05.B noted that the lack of C++ templated data structures could exclude applications where significant numerical precision was required.

- Participant P05.B noted the lack of asynchronous API calls for progress notification in the ML model training.

- Participant P05.B's use of much larger datasets than anticipated (raw audio data sets) uncovered bugs in the RAPID-MIX API implementation.

Participants expressed a desire for additional and more effective documentation about general machine learning concepts (P01.B and P03.B), specific API methods, and more domain-specific code examples (i.e., IML applied to audio). Participants also suggested additional features for the JUCE RAPID-MIX module such as:

- incorporating more types of ML algorithms, particularly for temporal modelling, such as dynamic time warping (DTW).
- providing more granular control over ML algorithms and evaluation methods by exposing more parameters for expert use (e.g. changing the architecture or activation function of neural networks).
- providing more ways to examine the ML models, for example through data visualisation, to aid understanding of model decision boundaries or model behaviour in higher dimensional spaces.
- improving the training speed of ML models.
- validation of input data for both training and model evaluation.

5.7.3 Two-week summer workshop with creative developers at eNTERFACE'17

This section describes another selected UCD action that investigated the usability and appropriation of a different subset of the RAPID-MIX API, the RapidLib.js, by a different types of users, creative coders. This UCD action took place in month 29 of the RAPID-MIX project. As with the previous UCD action, I wanted to investigate the following questions:

- What are the needs, goals and values of this user group?
- How do these users use the RAPID-MIX API and for what? What about machine learning or the RAPID-MIX API was confusing for these users? What errors did they make? What was unexpected?
- What other API features might they need, and which would they need the most?

We ran a two-week workshop at eNTERFACE, a yearly summer workshop organized by the SIMILAR European Network of Excellence. eNTERFACE 2017 was held between 3–15 of July, at Universidade Católica Portuguesa in Porto. This workshop followed the official beta release of the RAPID-MIX API (May 2017). The beta release included several new features; additional learning algorithms, such as DTW, Gaussian Mixture Models (GMM), Hierarchical Hidden Markov Models (HHMM)—via improved integration with the XMM package (Françoise, Schnell & Bevilacqua, 2013)—and particle filtering—via integration with Gesture Variation Follower (Caramiaux, Montecchio, Tanaka & Bevilacqua, 2014); a new class library with signal processing primitives (e.g. circular buffer, Root Mean Square, Mel Frequency Cepstral Coefficients, first- and second-order derivatives, etc.), and an improved web API for cloud-based multimodal data storage and retrieval.

This UCD action targeted creative coders who had a more diverse set of interests (i.e., not just audio programming), and who used a wider variety of programming languages and tools. It focused on helping participants to gain practical experience with elements of the toolkit, and on simultaneously identifying usability issues, learning obstacles and intended uses.

Thirteen participants (2 female, 11 male) with prior background in creative coding and multimedia were recruited in two rounds through research mailing lists, creative communities on Facebook, and personal contact networks. Most participants had Master's level degrees; three participants were PhD students; two participants were professionals in web development and game development, respectively.

In the weeks before the workshop, we surveyed participants about their background and motivations for attending. We then refined the UCD action plan to reflect these. For instance, as participants all had prior experience in JavaScript (JS), we narrowed the scope of evaluation to the JS subset of the RAPID-MIX API. The RAPID-MIX API JS library had been previously integrated into CodeCircle (Zbyszyński, Grierson, Yee-king & Fedden, 2017), an online live coding environment for beginning coders and computing students. CodeCircle aims to support efficient experimentation and prototyping activities (Parkinson, Zbyszyński & Bernardo, 2017; Zbyszyński, Grierson & Yee-king, 2017).

Each day of the first workshop week began with a researcher-led induction session. These sessions progressively introduced participants to ML concepts and use cases, and to the relevant components of the RAPID-MIX API. After the induction session, participants spent 2–3 hours engaged in hands-on exploration with the tools (Figures 3a and 3b). Each day concluded with a video-recorded group discussion.

In the second week, the workshop format changed to mentored project work. Participants worked independently on their creative projects, and at the end of each day a group discussion took place in which they reported on their progress and challenges. Participants also completed questionnaires after the workshop, in which they provided information about their experience with the different elements of the RAPID-MIX API and how they benefited from them.

We provided several resources for assistance, reference and learning during the workshop. The RAPID-MIX API code repository and website provided documentation. A Slack channel supported Q&As with remote mentors and participants. We also provided CodeCircle documents exemplifying how to use different functionalities of the JS RAPID-MIX API (e.g. classification, regression, temporal classification) with different sensors (e.g. mouse, webcam, LeapMotion[8], MYO[9],

[8] Leap Motion sensor, https://www.leapmotion.com/

[9] Thalmic Labs MYO, https://www.myo.com/

Figure 5.12: a) and b) Participants working on their projects and c) presenting a final project

Gametrak, etc.), and audiovisual outputs (e.g. WebGL[10], Web Audio API[11], P5.js[12], Three.js[13], etc.).

The workshop was structured to allow flexible participation schedules. Of the 13 participants who attended most of the first week, only the 5 participants (3 males, 2 females) who had previously enrolled for the full 2-week workshop attended the second week. The participants who opted not to carry on lacked availability due to academic or professional commitments and stated they had fulfilled their initial goal of getting a cursory understanding of ML and the RAPID-MIX API.

5.7.4 Results

At the end of the second week, four participants submitted their projects (as source code in GitLab[14] or in CodeCircle documents) and delivered a final demonstration to the remaining group. Participants presented the following projects:

- Participant P01.C explored the use of ML to exert more expressive control over the feedback loop and slide transitions of a Kodak carousel projector. She mixed and mashed-up CodeCircle examples until she focused her exploration on using ML regression with the Myo sensor's EMG and motion signals, to control different visual outputs such as colour gradients and animations. She also employed temporal classification with DTW, using an Arduino microcontroller board to control the projector (https://vimeo.com/225762966).

- Participant P02.C submitted two projects: 1) a web application which implemented the rock-paper-scissors game—single player against computer—for which he used RAPID-MIX API JS and Leap Motion for classification of hand poses, providing both pre-trained pose models and optional customisation features; and 2) a "Gesture server" application that used a server-side component to enable gesture recognition with the accelerometer data of a wirelessly connected smartphone (Figure 3b).

[10] WebGL, https://www.khronos.org/webgl/

[11] Web Audio API, https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

[12] P5.js, https://p5js.org

[13] Three.js, https://threejs.org

[14] Issues on Goldsmiths Gitlab RAPID-MIX Repository, http://gitlab.doc.gold.ac.uk/rapid-mix/RAPID-MIX_API/issues

- Participant P03.C used RAPID-MIX API regression with Leap Motion hand pose data to train and control 3D mesh deformation, iterating from using a simple regression model to control single-parameter mesh transformation, to employing a more sophisticated solution that used multiple models to control individual vertices (Figure 3c).

- Participant P04.C decided to build his own toolkit with building blocks for visualization of the Myo sensor data and ML processing, wrapping the RAPID-MIX API in a single-page web application. His project evolved from building client-side to server-side ML training and processing. The project involved recent web technologies but unfortunately was not finished before the conclusion of the workshop.

In the discussion groups, we asked participants to identify compelling uses of the RAPID-MIX API for creating future technology. Identified uses include: making products for others (e.g. games, physical activity recognisers, interactive music performances, smartphone sensor apps); creating personalised experiences for oneself; enabling social and group interaction; emotion classification; providing corrective user feedback; enabling more natural and expressive interaction and controllers; using it as a teaching material for kids in hands-on workshops; using it for efficient workflows and fast results when working with sensors; and combining different ML algorithms for simultaneous use in real-time applications.

User feedback about API usability was distilled from the group interviews and from the post-workshop questionnaire into the following items:

- All participants were enthusiastic about the speed of prototyping with the RAPID-MIX API, particularly using CodeCircle and the RAPID-MIX JS library. They found CodeCircle examples useful for prototyping interface designs with different sensors, and for providing building blocks for quick integration into their own creative projects.

- Participants appreciated the code clarity, simplicity and terseness of the examples provided. However, two participants (P02.C and P04.C) perceived the significant boilerplate code and poor-quality code comments negatively, mentioning that these impeded their understanding of the API. Other participants also (P03.C and some of the early first week participants) requested that we added better code comments to explain the role of constant values which were not clearly contextualised or explained.

- Participants P01.C, P02.C, P04.C complained about the examples' focus on audio, and the lack of examples applying ML to visual media. In general, the group was not knowledgeable about digital audio and found audio examples too abstract. Code examples that provided richer audiovisual feedback and control were most highly regarded.

- Participant P01.C and a few first-week participants suggested the provision of complementary high-level documentation which could give them a quick and broad explanation about ML concepts and expected API workflows. They also requested improvements to the structure and visual presentation of documentation, claiming it was not uniform across the whole set and that it was confusing to navigate.

- Most participants revealed difficulties in understanding how to use data with IML. Participant P01.C found it difficult to understand the conceptual difference between using data for training and for running ML models; she overcame this difficulty by creating different variables to store the datasets for each functionality and testing the outcomes step-by-step, offline. Another participant (P02.C) was not getting the expected classification results, because he had made a conceptual error of implementing training with raw data, and testing with RMS-smoothed (Root Mean Square) data.
- As in the JUCE Hackathon, some participants (P02.C, P03.C and P04.C) noted the lack of asynchronous API calls for progress and termination notification in the ML model training.
- All participants found the thread-hogging behaviour in the ML model training function problematic for browser applications. Two participants (P02.C and P04.C) opted for a server-side ML design implementation because this limitation made their application unresponsive.
- In general, participants expected a community forum, which did not exist at the time of the workshop. They suggested it would have been useful to collect their interactions in the workshop and to support future adopters and users of the RAPID-MIX API.

The data collected from the workshop included:

- my observation notes
- Slack channel logs
- pre- and post-workshop questionnaires
- 45 video recordings including:
  - group discussions of the first 6 days of the workshop (totalling 248 minutes of video)
  - final project presentations (4 videos with duration between 12 and 30 minutes each, totalling almost 80 minutes) which are stored in RAPID-MIX video repository
- 13 instances of source code of the participants' projects including CodeCircle documents and forks and git repositories

All the data was made available internally to the RAPID-MIX consortium. Due to the high volume of the primary data and the lack of budget and human resources, I was the only researcher carrying out data analysis. I employed a "lightweight" approach to data analysis, which consisted of examining each element of the primary data in two rounds, taking notes of what I considered the most important takeaways to inform design and development. There were no transcriptions of group discussions. Instead I used a naturalistic approach, in which I took notes and extracted links to the exact moments of the video recordings. These resulting links and notes were reported as findings on an internal report that circulated the consortium and also contributed to the writing of deliverables. These were validated by the co-authors and the internal reviewer.

From the user feedback collected during these action, I distilled a set of recommendations to inform further development of the RAPID-MIX API and its documentation (Table 5.2). These recommendations were validated internally by the Goldsmiths team. They were reviewed by the authors of the deliverable D3.2 (Bevilacqua et al., 2015) and published. In the next section, I provide further detail and discuss these findings, as well as the internal recommendations which they have led to in the RAPID-MIX API development. .

5.7.5 Discussion

The following sections discuss the main findings which emerged from the two lightweight UCD actions and which were grounded on observation and interviews with users about their experience with ML developer tools. We uncovered different needs, goals, expectations and challenges that different user groups faced using one particular subset of the RAPID-MIX API. We were also able to obtain more information about the users to support a deeper characterisation and definition of user archetypes. The artefacts that the users produced in our interventions also contributed to a better understanding and delimitation of the design space.

5.7.5.1 Uncovering the ML-developer experience with lightweight evaluation of the RAPID-MIX API

UCD actions for lightweight formative evaluation provided a better understanding of the developers' experience with the RAPID-MIX API. We found out about developer needs, goals and expectations, and challenges they experienced, developing and rapid prototyping with the RAPID-MIX API. Participants in the two UCD actions for lightweight evaluation of the RAPID-MIX API differed most prominently in their domain specialism, programming experience and software development skills. We obtained a distinct set of findings in each UCD action from each user group and we also found common needs and issues across them.

We found that most developers, particularly those at the low end of the spectrum of technical expertise, expressed needs related to ML conceptual knowledge. These were mostly framed in terms of learning resources which could more effectively support the acquisition of ML concepts. Additionally, developers expressed needs for learning resources related to the application and practical use of ML, particularly of code examples which could bridge ML concepts to an implementation in an intuitive and compelling way.

I organised findings according to two main categories related to RAPID-MIX API—usability issues and documentation issues. Documentation here can be understood as documents that provide ML conceptual knowledge and other resources accompanying the API, which provide practical knowledge for implementation, such as guides, tutorials and code examples. These findings led to a set of recommendations which I formulated and submitted to the public Gitlab repository as Issues to inform the development and refinement of the RAPID-MIX API. These recommendations and usability issues are listed in Table 5.2 and Table 5.4 respectively, at the end of this discussion section.

Audio software developers at the JUCE ML hackathon were positioned at the high end of the spectrum of technical expertise. Mostly, they employed the RAPID-MIX API in building applications related to their professional domain, audio software. Some audio software developers also took the opportunity to extend pre-existing projects, which they had been developing and for which they considered the integration of the IML workflow with RAPID-MIX API a good fit with their goals. Audio software developers expressed need for more power and expressiveness from the ML developer tools. These needs were framed in terms of additional precision, flexibility, granularity of control and parameterization, and speed in model-training. They also required additional learning resources. However, contrastingly to the other less skilled developers they asked for advanced and focused resources such as the technical descriptions of the underlying algorithms. These findings led to my recommendation R05 "*Providing further resources to the different underlying algorithms*" (Table 5.2). They were also critical about lack of advanced features, such as data validation in API methods, and lack of feedback through error reporting.

We also verified that a basic set of documentation resources was not an impeding problem for audio developers using the RAPID-MIX API at the JUCE ML hackathon. They "managed to get by" with a brief introduction to the RAPID-MIX API at the beginning of the session and with these basic learning resources. Rather, audio developers were very autonomous in the exploration of the RAPID-MIX API, and proceeded to probe the potential of the RAPID-MIX API with simple test projects or with the provided examples. Mostly, they inspected the API source code, glanced over the documentation, and quickly advanced to the development of their proof-of concepts, or extension of their pre-existing concept(s). One of the audio developers was so autonomous and aware of his specific needs, that he "altered" RAPID-MIX API on his own by stripping away the wrapper layer that adapted it to the JUCE client framework, in order to run it in his embedded hardware setup.

Creative coders at eNTERFACE'17 attended extended sessions about ML concepts with follow-up hands-on sessions, throughout the first workshop week. Nevertheless, they faced difficulties in understanding ML concepts and related programming concepts including data structures, data and features, and auxiliary development tools, such as code repositories and protocols. Interestingly, creative coders at eNTERFACE'17 were more vocal about the basic requirements and simple features than those of their counter parts at the JUCE ML hackathon.

In the group discussions at eNTERFACE17, creative coders expressed the need for learning resources and documentation elements which could provide the big picture of ML. For them, there was no element of RAPID-MIX API documentation which provided a broad, encompassing per-

Table 5.2: Recommendation for improvement of the RAPID-MIX API based on the empirical findings

	Conceptual knowledge
R01	Providing the big picture of ML concepts and how they relate to RAPID-MIX API
R02	Communicating the Interactive Machine Learning workflow to users
R03	Providing an incremental approach for learning how to use the API
R04	Explaining the relation between inputs, classes, outputs and mapping
R05	Providing further resources for the different underlying algorithms
R06	Explaining the concept of “background class”
	Examples
R07	Writing examples as building blocks
R08	Writing the examples code for learning and understanding
R09	Providing contrasting examples
R10	Segregating boilerplate code
R11	Providing design patterns for applying the RAPID-MIX API
R12	API methods for training-data quantification
	Documentation
R13	Regression requires additional explanation on the website
R14	Self-explanatory code and commenting practices
R15	Refactor variable names in code snippets of the website
R16	Introduce supporting technologies (git, terminal, node, etc) and curate links to third party resources

spective of ML concepts. They suggested the inclusion of visual and informational elements such as mind maps, diagrams, or cheat sheets in the RAPID-MIX documentation. This led to my recommendation R01, “*Providing the big picture of ML concepts and how they relate to our API*” (Table 5.2) for API resources. In the contextualisation of this recommendation, I suggested designing mind maps and interactive instructional elements based on Kohavi and Provost’s 1998 glossary of terms, an academic resource for ML concepts and definitions, which is considered both comprehensive and simple, and which I briefly reviewed and contextualised on Section 2.1.1.1.

Creative coders at the eNTERFACE’17 were mostly developers at the mid or low end of the spectrum of technical expertise. They had more limited knowledge and programming experience than audio developers at the JUCE ML hackathon, and provided fundamentally different feedback. As a whole, creative coders showed less narrow focus in a specific domain, but were also driven by personal interests. They also appeared to be more reliant on RAPID-MIX code examples, which they scavenged to build their own projects. Mainly, creative coders expressed needs for two types of resources: i) learning resources that could support a rapid and effective acquisition of ML conceptual knowledge, and ii) practical examples which could provide them with practical

Table 5.4: The Adapted Cognitive Dimensions framework used in our study

Major usability issues	
UI01	Lack of progress and termination feedback in the ML model training
UI02	Thread-hogging behaviour/UI-blocking when training a classification/regression model
UI03	API documentation fragmentation
UI04	Server-side technologies are not packaged for end-users, require a complex installation and suffer from bugs

knowledge in the application of ML.

On the same note, creative coders at eNTERFACE'17 expressed the need for resources that could help them relate the fundamental ML concepts with a practical implementation that the RAPID-MIX API could support. In particular, creative coders suggested that the documentation explaining relation between how inputs (or features) and outputs (or labels) are set was cumbersome. We found that this need actually reflected a set of related misunderstandings and mistakes that participants using the RAPID-MIX API experienced in both UCD actions. One common misunderstanding which supported this assertion was verified in the case of code examples with interactive applications that made use of "composite labels". This creative pattern consisted of a many-to-many mapping that associated multiple regression outputs to multiple parameters of a synthesizer. This pattern could be considered more unusual in the domain of interactive audio applications, and perhaps slightly more convoluted.

There were indeed complaints from users at eNTERFACE'17 that RAPID-MIX API was opinionated concerning the domain of application to audio. However, the "composite labels" pattern also showed that, in terms of ML code integration and its comprehension, such an example—a simple aggregation of ML models for supporting a many-to-many mapping—without a solid understanding of basic concepts, led to misunderstandings at both the conceptual and implementation levels. "Composite labels" could be understood as a creative pattern which originated from the domain of audio software and that challenged the comprehension of the creative coder who did not have that background. However, this misunderstanding seemed related with another common misunderstanding we found with audio developers at the JUCE ML hackathon.

These issues led to my recommendation R04 "*Explaining the relation between inputs, classes, outputs and mapping*" (Table 5.2). We also found that IML workflow is difficult to convey with static sources of descriptive knowledge, such as text (documentation, comments, etc.) or even code. This is due to procedural nature that the workflow entails, and live demonstrations work very well. In recommendation R02, "*Communicating the Interactive Machine Learning workflow to users*" (Table 5.2), I proposed new ways of complementing the documentation with richer media, such as video and animations, which are more effective at communicating the procedural nature of the workflow and yield better adoption.

These issues showed that it is critical to provide support for an incremental approach to ML. Users needed guides starting with minimal examples of fundamental ML concepts, such as classification and regression, their RAPID-MIX API implementations as class library objects and small training data sets and the RAPID-MIX API, showing how to implement basic ML code. An incremental approach would build up progressively from the 'sandbox' that these basic code examples provide, towards a more compelling use case grounded on real world scenarios.

This led to recommendation R03, "*Providing an incremental approach for learning how to use the API*" (Table 5.2). In fact, users suggested structuring code examples with a narrative style, based on "recipes" or short stories and grounded on specific use cases. The "Nature of Code" was given as a reference for providing an effective incremental approach to teach multimedia software development. These findings call for rationale in the design of progressive instructional and pedagogical approaches, which may start with the exploration of the fundamental concepts of machine learning and gradually develop complexity. It also calls for rationale in identifying design patterns for ML and in communicating them as such.

At eNTERFACE'17, there was also consensus in the group discussion about participants' perception about how the documentation of the different components was fragmented and of the impact of this issue. At the time, some of the components from different RAPID-MIX consortium providers had different websites, where content had a a distinct graphic design style and user experience. This made obvious the different origins and authors of the resources. In the group discussion, participants discussed and agreed that there should be a common medium, method and a uniform style for documentation. They suggested that this would reduce potential friction in adoption of the RAPID-MIX API. Some of the participants provided examples of developer tools with inspirational documentation and learning resources, such as ORM for MySQL, Postgres, loadash, underscore, KNX, and Processing.

So that these recommendations could directly influence subsequent development, I created Gitlab Issues (i.e. Gitlab the adopted development management platform) within the RAPID-MIX API code repository. Table 5.2 lists the recommendations organised in API-related categories—i.e. ML conceptual knowledge, code examples, documentation. Table 5.4 lists major usability issues. Each item in these tables is linked to a corresponding issue in the public Gitlab repository. Each item comprises a contextualisation for observations and how they were obtained in the UCD action, and a rationale for objective recommendations to improve specific API aspects.

5.7.5.2 Understanding the user

UCD actions in RAPID-MIX contributed to a better understanding of the needs, goals and values of the target users of the RAPID-MIX API. The JUCE Machine Learning Hackathon (Section

5.7.1) focused on understanding audio developers, users of the JUCE RAPID-MIX module which wraps the RAPID-MIX C++ API. The eNTERFACE17 workshop (Section 5.7.3) was useful for understanding a more diverse user group—creative coders with different skill sets—and how they used the JS library. According to Cooper, Reimann and Cronin (2007), design personae can support more natural and effective reasoning about design. These two actions motivated the definition of design personae (Clarke, 2007; Cooper et al., 2007) as part of a set that characterises the users of RAPID-MIX API. In creating these personae, I wanted to ground the design process in the most precise user population as possible.

- Miguel, 35, is an experienced audio software developer with a CS degree. He programs mainly in C++ and uses JUCE as his primary development tool (most participants at JUCE ML Hackathon). Miguel owns/works for a start-up that produces VST plugins and mobile music apps. He is interested in machine learning but does not really know what it is about, beyond data mining in large databases and music information retrieval. He thinks ML processes are a time-intensive and boring process which someone else should take care of. He recently heard about ML for rapid prototyping at a conference, and was amazed by the speed that it could be used to prototype certain complex functions. It might be a great tool for his prototyping kit. As a professional developer, Miguel has experienced many APIs and quickly develops a critical perspective about them, concerning their performance, overall design and architecture. He likes to do a quick overview to grasp the concepts, leap into the documentation and API reference, and then look through the examples. He quickly begins implementing some wacky throw-away idea using problem solving techniques (divide and conquer, trial-and-error iterations, etc). As a systematic developer (Clarke, 2007), Miguel builds deep technical understanding and prides himself on developing DSP code with maximum performance and minimal memory footprint.

- Jill, 23, is a creative computing and media student. She is interested in creating and expressing and is driven by concepts more than by technology. Jill learned basic coding skills in C++, Python and JavaScript and has experimented with Wekinator in the class. She has experimented with physical computing, biofeedback sensors, and has programmed a couple of simple games in Unity. As an opportunistic developer (Clarke, 2007), she writes code in an exploratory fashion and develops the sufficient technical understanding to solve her design problem.

- Kwame, 28, is a musician, an electro-acoustic and audio-visual composer. He works mainly with Max/MSP and other data-flow programming environments. He has a passion for building interactive installations, delivering expressive performances and for developing new interfaces for musical expression. He has been commissioned to write interactive operas and contemporary dance performances and works with composers and choreographers who value fast results. He has used Wekinator to map motion sensors to his Max/MSP patches for real-time gestural expression, and he finds it really quick to use. He also attended Rebecca

Fiebrink Kadenze MOOC to understand better how ML algorithms work and how to apply them to creative projects. However, he finds Wekinator challenging to use in more complex scenarios, where a mixture of different types of ML models are required, mainly for temporal data. As a visual programming developer, he wants to apply the IML workflow with greater flexibility using components from his data-flow programming environment.

- Kalifa, 28, is a web designer and developer. She has a degree in Web Design and has advanced skills JavaScript and Node development. She has multiple small pet projects, including contributing to OSS projects with repositories on GitHub. She strives to keep up with the fast pace of web development technologies and workflows. She is very interested in the future of Web Audio and in web innovation. She is curious about all the buzz around ML, but since she sees it mostly as server-side technology, has not really tried it. But she does not really know how it works and what to use it for, particularly with client-side web technologies, which has not been an environment in which the interesting things with ML are developed and showcased. She deals with a lot of component and service APIs and really values a good experience with development workflows, APIs and documentation. In reality she uses documentation mostly for troubleshooting and overcoming walls; she really prefers good examples, tutorials and resorting to community support on StackOverflow. She is also intolerant with issues that break the experience and the responsiveness of web apps such as long waits, glitches, snaps, and bad typeset choices. As a pragmatic developer (Clarke, 2007), she follows best practices and strives to develop sufficient technical knowledge of a technology to use it robustly.

- Karel, 35, is a hobbyist and self-proclaimed maker and tinkerer, uses creative coding platforms such as P5, OF, and embedded hardware, such as Arduino and RaspberryPi, and is often doing hackathons. He has a day job, but he really loves to tinker with new sensors and hardware. He has published on Instructables and got some community buzz. He dreams of coming up with an idea and implementation of a truly useful and successful IoT product, and step into entrepreneurship. He knows about ML from IBM Watson, but he has never really used it because of all its layers of infrastructure, and how it is detached from grassroot products. As an opportunistic developer (Clarke, 2007), he writes code in an exploratory fashion and develops the sufficient technical understanding to solve his design problem.

These personae were created through the amalgamation of real user characteristics to illustrate the breadth of software development skills, experience, motivation and technical approach expected from creative and music technology developers. These personae communicate the knowledge about the main target user groups of the RAPID-MIX API which was uncovered through the UCD actions. This knowledge concerns the main user groups' needs, goals, expectations and challenges, as well as typical tasks and context.

For instance, Miguel, the audio software developer persona which I developed based on my observations, interaction and interviews with audio developers—at the JUCE hackathon and follow on contacts, e.g. participants P04.A, P01.B–P05.B—represents a user group of RAPID-MIX API users at the high end of the spectrum of technical expertise and experience. This groups includes professional developers with advanced development skills and more systematic approaches to programming. As with many users represented by this persona—who might have had a brief contact in their education with more traditional flavours of ML—he is inspired by the support for rapid prototyping and fast results that the RAPID-MIX API supports. Nevertheless, given the level of performance and robustness that users characterised by such an archetype might demand, it may be reasonable to expect them to require other ML tools to deploy models in production.

Besides the characteristics uniquely conveyed by these personae, we found additional characteristics that were transversal to the main groups. Most importantly, the user groups shared having little or no experience in ML. The lack of ML proficiency was a common characteristic to both users of the RAPID-MIX API as proxy-users and their end users. Further, these groups shared other characteristics—e.g. a high degree of intrinsic motivation, customisation and development skills, anticipation of market needs, building for their personal needs, hobbyism, bricoleurism—that have been captured by previous research on Lead Users (von Hippel, 1986) and End-User Developers (Lieberman et al., 2006).

This knowledge about the user has provided a frame for the design process and has had practical implications. For instance, after considering the needs of the audio developer, we exposed lower-level primitives for configuration of the neural networks in the RAPID-MIX API C++ (e.g. hidden layers, activation function, etc); we built templated data types into the library to allow a finer control over the numerical precision required in many audio applications and embedded hardware. We also added additional ML algorithms such as DTW to the library, based on the overall interest in temporal data. To address the needs of the creative coder, we created new examples with more visual feedback and better code styling. Online documentation was restructured to integrate a set of interactive tutorials with increasing complexity and contrasting features onto the website; these changes aimed to provide a smoother learning curve and an accelerated learning experience to the general audience.

5.8 RAPID-MIX API usability evaluation with the Cognitive Dimensions framework

The Cognitive Dimensions (CD) is a framework for broad-brush usability assessment of information structures (e.g. programming languages, application programming interfaces, notations) (Green, 1989; Green and Petre, 1996). It can provide a brief and high-level description of design trade-offs and usability problems while avoiding "death by detail" (Blackwell and Green, 2003, p. 106).

The CDs framework is intended to support experts discussing and judging the design of systems that are design tools. The Cognitive Dimensions provides system designers with a framework and terminology for addressing usability issues, of notations or interaction languages, and how well they support the intended user activities, based on experts' needs or guessed user needs (Blackwell and Green, 2003). It can be used to measure the discrepancy between user expectations and participant performance in programming tasks.

The CDs framework has been used before in the analysis of interaction design for visual programming (e.g. Max/MSP, PureData) and text-based languages (e.g. Csound, SuperCollider) for algorithmic composition (Bellingham et al., 2014). The CDs help with the analysis of specific structural properties of notations and cognitive tasks of employing notations. The list of dimensions as initially defined, consists of:

- Viscosity: resistance to change
- Abstraction: types and availability of abstraction mechanisms
- Hidden Dependencies: important links between entities are not visible
- Premature Commitment: constraints on the order of doing things
- Secondary Notation: extra information in means other than formal syntax
- Visibility: ability to view components easily
- Closeness of Mapping: closeness of representation to domain
- Consistency: similar semantics are expressed in similar syntactic forms
- Diffuseness: verbosity of language
- Error-Proneness: the notation invites mistakes and the system gives little protection
- Hard Mental Operations: high demand for cognitive resources
- Progressive Evaluation: work-to-date can be checked at any time
- Provisionality: degree of commitment to actions or marks
- Role-Expressiveness: the purpose of an entity is readily inferred

The CDs framework was extended with a general questionnaire which offers the CDs definitions to the respondents, allowing them to evaluate a system they have used according to features of their preference (Blackwell and Green, 2000).

5.8.1 Method

The overall objective of this study was to obtain a deeper understanding about how the design decisions and trade-offs of an API for rapid prototyping of creative technology with IML affect its usability and developer experience. For the purpose of the study, we refined this objective into the following key research questions:

1. What supporting and contradictory evidence do we find that the RAPID-MIX API is usable?
2. How do users perceive the API design trade-offs and how do they affect usability and developer experience?

To answer these questions, we designed a study with participants who used RAPID-MIX API in their work and were asked to report on their experience using an adapted version of the Cognitive Dimensions framework questionnaire by Clarke (2010). The questionnaire answers were analysed using a qualitative approach.

Participants

We selected participants who had used at least one subset of RAPID-MIX API within a creative software project. Participants signed a consent form to participate in the study. Our sample set of 12 participants (1 female, 11 males) comprised 6 professional developers working in 3 small and medium-sized enterprises (SME) in creative technology, and 6 creative non-professional developers creating systems for their own personal use (students of different levels, spanning undergraduate, master level and PhD students; see Table 5.6).

Participants had varying software development experience, varying ML experience (from none at all to some experience with frameworks such as tensorflow, scikit-learn, etc.). Participants had used different subsets of the API (RapidLib C++, RapidLib JS, XMM C++ or mano-js) for varying amounts of time (from less than one month to a little more than a year). Some participants used the API in personal projects or proofs-of-concept outside the commercial sphere; others developed for commercial purposes in a professional context.

Personal projects included interfaces for creative performance where the IML was used to create mappings from various sensors and signals to multimedia control parameters. For instance, P01.D used IML to create a system in which input data from a Myo (Thalmic Labs Inc.) sensor armband drove outputs that simultaneously controlled lighting, visuals, and sound through Ableton Live (Ableton); P11.D developed an audiovisual synthesizer that used regression to establish a tight connection between sound and live graphics.

Commercial products created by participants included: a JS front-end component that integrated the IML workflow into a commercial biosignal analysis toolkit for rehabilitation engineers working with patients (P03.D, P04.D); an intelligent drum sequencer for iOS with custom gesture activation (P05.D, P08.D, P12.D); and a software-as-a-service that coordinates collective events, such as movement workshops and soundwalks, using the multimodal and multimedia capacities of attendees' mobile devices (P09.D).

The Cognitive Dimensions questionnaire

We employed an adapted version of the CD framework questionnaire (Clarke, 2010) as our research instrument. This questionnaire has been developed to support a comprehensive and systematic understanding of participants' experiences with the API, broken across several different dimensions. Clarke's questionnaire provides several benefits over the original CD questionnaire, as it is tailored for API evaluation, and it also has an increased focus on learnability (i.e. introducing additional dimensions including Learning Style, Penetrability) (Clarke, 2010).

We were inspired by prior studies which fine-tuned Clarke's questionnaire to specific domains—e.g. Watson (2014) introduced high-level groupings of dimensions for a more effective distillation of results to improve API documentation planning, which we adopted. Wijayarathna et al. (2017) aimed to evaluate aspects of their API that were specific to cryptography by introducing additional dimensions (e.g. End-user protection, Hard to misuse, Testability). We have adopted two dimensions, Testability and Error-proneness, the former from the questionnaire of Wijayarathna et al. (2017), and the latter from the original CDs questionnaire (Blackwell and Green, 2003) with minor changes. Our full questionnaire appears in Appendix H. Table 2 summarises the 14 dimensions used in our questionnaire, grouped into four high-level themes. Each dimension was addressed by several questions (Appendix H).

We first delivered a pilot version of our questionnaire to two participants (P03.D and P04.D). This version was longer and closer to the original version by Clarke (2010), and we received complaints about its length. We therefore shortened the questionnaire by removing some redundancies. The final questionnaire was delivered online, on paper, or through in-person or remote interviews, due to the geographical spread of the participants.

Table 5.6: Listing of study participants and their API client software

ID	Software Development Experience (years)	ML Experience	API subset used	Time using API (months)	Use (personal, commercial)	API Client Software
P01.D	4	some	RapidLib C++	8	personal	Openframeworks application with IML for sensor-based choreography
P02.D	1	some	RapidLib C++	11	personal	Max/MSP external and toolkit for musical therapists building bespoke digital musical instruments for children with special needs and disabilities
P03.D	6	some	RapidLib JS	1	commercial	*[a] IML component for a platform-as-a-service for rehabilitation engineers using BITalino biosignal data
P04.D	6	some	RapidLib JS	6	commercial	*
P05.D	14	some	XMM C++	>12	commercial	** Mobile iOS app and test suite for intelligent drum sequencing based on IML motion data and IML
P06.D	5	some	RapidLib C++	5	personal	Openframeworks toolkit for desktop and Android for sonification of BITalino and smart watch motion data with IML
P07.D	3	some	RapidLib JS	<1	personal	Interactive web-based tutorial for regression and IO mapping with IML
P08.D	5	none	XMM C++	1	commercial	**
P09.D	5	none	mano-js	<1	commercial	Platform-as-a-service for supporting body movement workshops and soundwalks with mobile motion data and IML
P10.D	1	some	RapidLib JS	<1	personal	Browser-based React application for structuring data and IML pipelines
P11.D	1	some	RapidLib C++	>12	personal	Openframeworks sensor-based reactive audiovisual installation with IML
P12.D	7	none	XMM C++	6	commercial	**

[n]* and ** refer to one project at a different SME, which the participant worked on

Table 5.8: The Adapted Cognitive Dimensions framework used in our study

Learning	
Abstraction Level	Overall magnitude of abstraction and style of abstractions of the API
Learning Style	Learning requirements and style encouraged by the API
Penetrability	Ease of access, retrieval, exploration, analysis, and understanding of the API components
Understanding	
Consistency	Similar semantics are expressed in similar syntactic form
Role-expressiveness	Purpose of an API component is readily inferred
Domain Correspondence	Clarity of domain mapping of API components
Usage	
Working Framework	Size of conceptual chunk or amount of context necessary to work effectively
Elaboration	Extent to which API must be adapted to meet developer needs
Viscosity	Resistance to change in refactoring
Premature Commitment	Constraints in the order of implementing API code
Error-proneness	Error incidence, recoverability and support
Application	
Work Step Unit	Amount of programming task completion achieved in a single step
Progressive Evaluation	Work-to-date can be checked at any time
Testability	Types of evaluation and assessment metrics that are adopted

5.8.2 Results

In this section, we report our findings about each of the dimensions included in our questionnaire. We employed content analysis using NVivo to analyse responses. We adopted a deductive analytical approach in which we used codes based on the CD framework, on the higher-level themes in table 2, and on an auto-encoding analysis performed with NVivo.

We also tried to find correlations between the variables Software Development Experience, ML Experience, API subset, and time using the API, in the closed-end questions of each dimension (e.g. Q1—perceived level of abstraction, Q8—learning experience, Q11—experience with amount of context, etc.; Appendix A). Given the size of our sample, we ran Pearson's chi-squared test with Yates correction, and Fisher's exact test. We found no support for contingency between those variables in the quantitative results in the dimensions, as none of the tests yielded statistical significance.

Abstraction level

Questions pertaining to this dimension aimed to investigate the appropriateness of the abstraction level of the RAPID-MIX API. We asked how appropriate the abstraction level was for participants' development needs and why (Q1, Appendix A), and whether participants felt they needed to know about the API's implementation details (Q2).

In responses to Q1, 7 of 12 participants found the overall API abstraction level 'just right', and 5 of 12 found it 'too high level'. No one found it 'too low level'. Five of the 7 participants who had used the API for longer than 2 months found the abstraction level 'just right'. Participants who used different subsets of the API differed in their responses; all the participants using RapidLib C++ (4 participants) or mano-js (1 participant) considered these had the right abstraction level, and 3 of 4 participants using RapidLib JS considered it too high.

Participants who found the abstraction level just right described how it enabled them to achieve their development goals (P01.D, P02.D, P06.D, P11.D). These included rapid prototyping ("...I was able to do rapid prototyping, rapidly!", P06.D), simple implementations ("for a quick and simple implementation the abstraction level works well", P03.D), and proofs-of-concept (P04.D). Participants also referred to the positive development experience the API provided, having found it "extremely easy to use in C++, which is usually a very confusing language" (P02.D), or non-obtrusive to the creative process—"I was able to implement most of the RapidLib functionality without losing my creative flow" (P06.D). P04.D indicated that the API "facilitates the final programmer use" and saved her a lot of time by avoiding having to handle implementation details.

Participants who found the RAPID-MIX API abstraction level too high (P03, P05.D, P07.D, P08.D, P10.D) experienced problems mainly because they needed further understanding of lower level details—"when I tried to learn a little more, knowing, for example, which model was being used, I saw the abstraction level as a hindrance" (P03.D)—or finer-grained control over certain API features—"I found that while the algorithms work, I would have liked a bit more control over certain algorithms" (P10.D). Some participants complained about the lack of transparency of RapidLib JS's high-level objects (Classification and Regression), which prevented them from knowing which algorithms were in use. Because RapidLib JS is transpiled from C++ to asm.js, the source code and algorithm implementation is more opaque than the C++ version.

Participants who stated that they needed to know the underlying implementation (Q2) presented different reasons for this. Three participants (P05.D, P07.D, P10.D) found undocumented algorithm parameters—e.g. k in the k-nearest neighbour algorithm, the number of hidden units in the hidden layers of the multi-layer perceptron—and needed to understand how to use them, and so had to look further into the implementation. Some of these participants worked on a product and related their reasons to needing a deeper understanding for customer-facing projects (P03.D, P08.D). For instance:

"I needed to know what was behind the object. If I am going to make a product or give a chance to a customer to use one of our solutions based on the result of the api, and for some reason, something wrong happens, it would be necessary to have a deeper knowledge of the whole object." (P03.D).

One professional developer, P05.D, considered the understanding of the underlying implementation vital—"The API is very concise and there's not much to learn; however choosing the correct parameters is a 'dark art'" (P05.D). One participant found the library opinionated ("it is intended to work in a specific manner") and had to look to the implementation to adapt it to their needs.

Participants who mentioned not needing to know the underlying implementation either mentioned that they already knew it, or, that they had the sufficient knowledge to be able to use the API successfully—"I felt that I needed general knowledge of how regression and classification algorithms worked. However, for my purposes this was enough. I could then just use the API without needing to know the exact implementation." (P11.D).

Learning Style

The questions about learning style aimed to determine what knowledge was essential to use the API successfully, how much new knowledge participants had to acquire, and how participants went about using API documentation to attain this knowledge.

Participants perceived knowledge of the following ML concepts to be important in facilitating use of the API (Q3): the probabilistic nature of ML (P05.D); the greater importance of the choice of data in comparison to the choice of algorithm (P01.D, P05.D); basic ML concepts, such as the difference between regression and classification (P02.D, P04.D, P11.D); the stages of the supervised learning workflow, such as collection and pre-processing of data, training and running the models (P01.D, P03.D, P04.D); and understanding the ML algorithms' implementation and inner workings (P01.D, P03.D, P07.D). They also identified the following knowledge of non-ML topics as useful (Q4): threading and non-blocking async architectures (P06.D), client/server architectures (P09.D), deeper programming language knowledge (e.g. using generics) (P12.D), statistics for modelling data (P05.D), and practical knowledge about sensors, Human-Computer Interaction (P11.D).

Participants' responses about their learning strategies (Q6) indicated that both novice and experienced developers tended to adopt an opportunistic approach (Clarke, 2007) to learning about the API: they frequently learned by copying sample code and employing hands-on exploration. The more experienced developers appear to have complemented this with a more top-down approach to learning about the API components or architecture.

The majority of participants (9 of 12) indicated that they had to learn "just [the] right" amount to use the API (Q8). These participants defended this response with answers that mentioned the simplicity, ease of use, and beginner-friendliness of the API. For instance, participants wrote that the "code of the API is simple and concise" (P05.D), that it was "straightforward to use without having to read too much documentation" (P07.D), and that "I didn't have to learn anything new to use the API and I didn't want to learn a lot to train such simple models" (P02.D). The other 3 participants stated that RAPID-MIX API documentation did not provide enough resources to support their learning, particularly regarding choosing appropriate algorithms and their parameterisations for a given problem. P12.D wrote "one is left guessing numbers and trial and error exploration, if there's no previous ML experience," and P01 wanted "more complex examples so that people can try different ML structures."

Working framework

Q10 aimed to elicit an understanding of the amount and type of information (i.e., 'context') a user needs to maintain or keep track of while working with the API. Eleven participants responded, choosing multiple elements from the list of contextual information (one participant did not respond). The top 5 classes of context identified as necessary by respondents were: API methods (10 of 11 participants), API classes (8), data structures for training and model input/output (7), database (e.g. JSON, XML, CSV, database management service) (7), and types of ML algorithm (6). Less common responses included: local scope variables (4 of 11), system configuration settings (4), app configuration settings (3), global scope variables (2), registered events (1).

When we asked participants to describe if this 'amount of context' was "too simple," "too complicated," or "just right" (Q11), 8 of 12 participants reported reported it was "just right". Participant explanations suggest this was driven by the simplicity of the API interface—"not very demanding, inasmuch as methods presented simple syntax. When developing, I didn't usually keep track of them. When problems arose, it was always easy to navigate to tutorial examples and spot where scope or method syntax was not correct." (P03.D). Contrastingly, the participant with the most ML expertise conveyed his reasons for what is important about context in more of a conventional rationale—"I needed to keep all the context in mind that is relevant to using the algorithm. I guess I don't care that much about the particular data structure that the API expects, so it would be nice to not have to think about that. I don't see how you could avoid that though" (P06.D).

Two respondents found that the amount of context they needed to keep in mind was too complicated. The less-experienced of the two found difficulties developing architectural support for an increasing number of model outputs—"adjusting the output parameters of my application took a bit of time and thought to figure out what parameters needed to be global and what parameters needed to be local." (P10.D). The other respondent, a more seasoned developer, implemented a non-blocking asynchronous threading architecture to make robust use of the API—e.g. "Training with large amounts of data can take a long time and should be non-blocking e.g. a future. However, it also needs to be cancellable." (P04.D)—which entailed the use of a more comprehensive and complex "context".

Interestingly, one participant referred specifically to training data and its specificities for the IML workflow as part of the 'context' to be kept in mind—"Often in the Application I would visualise the training data or rely on remembering it. So just being able to save the model without the training data was not useful and caused complexity issues. Especially when the training time of the models is very short and the datasets are small." (P02.D).

Work step unit

Q12 asked participants whether the overall amount of code they had to write to integrate the API into their projects was too much, too little, or just right. Eight of 11 participants mentioned that their experience was just right.

The remaining participants answered that they had to write too much code. Their explanations identified several tasks that appeared to require too much code: a) validation, b) data management, c) concurrency management, and d) management of model outputs. Three participants mentioned validation code as necessary to make their application safe and robust—e.g. "there's not too much error handling, or check on the data format/range" (P12.D). Two participants referred to concurrency—e.g. "not 'too much' code directly related to the API, just too much boilerplate wrapper code in order to use the API successfully in the context of a large multithreaded mobile app with GUI and audio" (P05.D). Three participants mentioned having used an extensive amount of code to create data structures and conduct data management in the application. For instance, "I had to write lots of code for formatting training data, I feel like the API could have given an interface for recording, building and editing data sets rather than needing to be given the whole dataset at once or relying on user-written C++ vector functions to edit training data" (P02.D).

Progressive evaluation

Q13 asked participants about the amount of work needed to evaluate progress in using the API. Notably, though participants knew the questionnaire was focused on evaluating the API itself, we found that the majority of responses related to the task of evaluating progress of the IML workflow outcomes (i.e., the quality of training data, the training process, and the model results) rather then just progress in establishing a functional pipeline.

Participants identified the simple API interface as facilitating progress evaluation —e.g. "It was very easy to evaluate the integration of the API with my application. Because of the simple user interface of the API, I knew exactly what to expect from each method of the API." (P02.D); "there is very little code required to use the API. Evaluating the performance of the tool, selecting the source data inputs, choosing a frame rate, ensuring orthogonality took up 10% of our time." (P05.D).

Responses that expressed difficulty in evaluating progress shared some common themes. For instance, respondents complained about the lack of integrated visualisation tools—"I evaluated my progress in using the API by implementing visualisations in D3 [...] I would probably like to minimise the amount of time spent on visualisation code" (P07.D). Others complained about the lack of functionality to provide feedback about model accuracy improvements—"There's no proper visualization of improvement, one is left with the trial and error to determine if the classification is improving or not, and no information on how good/bad it is.” (P12.D). One participant referred to the high abstraction level as a hindrance for progressive evaluation—“Because some of the functionality of the API is hidden for advanced or personalised use cases, I wasn’t completely sure about my own progress” (P06.D).

Premature commitment

Q14–Q17 examined how participants perceived the level of premature commitment required by the API—i.e., the need to make certain decisions too far in advance, and inflexibility in the order in which decisions had to be made.

Eight of 12 participants reported that they were forced to think ahead and make early decisions (Q14). Most participants found it necessary to make early decisions about data sources and preprocessing, data structures for inputs and outputs and their dimensionality, and the state machine architecture that supports switching between the modes of training and running the models (Q15). Some of the more advanced users, or users with more complex requirements for commercial product implementations, referred to planning the integration of the API components according to specific aspects of their use case—for instance, within a client-server or concurrent architecture.

Penetrability

Questions about penetrability aimed to understand the degree of ease with which developers could access information about API components, and explore, analyse and understand their working details in order to achieve specific development goals.

Eight participants encountered some difficulties in finding necessary information about API details (Q18, Q19), indicating that the documentation of API subsets was insufficient. Most of these respondents had, at some point, finer-grained implementation requirements, for which necessary details about the API became hard to find. Seven participants indicated having to learn about specific ML algorithms and parameter configurations (Q20). Some participants learned about these as they worked—e.g, “Online tutorial materials and examples were very helpful. However, should deeper potential of the API be explored, I can’t say that all questions would be easily answered.” (P02.D); “As my own knowledge [of IML] progressed I would have liked to be able to find out more detailed information about the neural network and how it performed the regression analysis” (P03.D).

Participants reported working to improve their understanding of the API (Q22) mainly through the process of trial-and-error exploration (5 participants) and by reading through the API source code (4 participants)—“Largely through trial and error I began to get a sense of how the regression model worked” (P08.D); “By breaking it, using intellisense to find functions that were templated to exist, but did not have implementations in some models, so I started reading more of the API’s source" (P01.D). Some participants reported needing to use direct communication with the API developers (P01.D, P06.D, P09.D) and resorting to external documentation resources (P05.D).

Four participants believed the API and its documentation provided enough information for their needs (Q23), found easy access to that information (Q19), and that there was no lack of detail (Q18). Most of these participants either had simple goals and remained at a high implementation level, or their exploration was technically-driven rather than design-driven—"My original interest [lay] in the C++ API, but resources and adaptation to the final product needs made me shift towards Javascript, which had magnific learning materials" (P04.D)". Some participants admitted not understanding the working details but were satisfied working with a 'black box'—e.g. "I didn't fully understand then. The results were adequate enough for our application" (P09.D); "I had no knowledge of the implementation details or how the API is generally structured apart from what was obvious from the code examples" (P05.D).

Elaboration

Q24 asked about the ways in which participants had adapted the API to fulfill their design goals (if any). Five of 12 respondents used the API 'as-is'. Five others reported having wrapped the API in adapter classes to add the necessary functionality to overcome specific limitations of the API. Three of these respondents had added error handling and support for asynchronicity.

Two participants reported having forked the API and changing the library file structure. One respondent hacked the API to improve the learning capacity of the default regression object. His hack approximated the functionality provided by an undocumented MLP parameter—"The hack was to increase the dimensionality of the input vectors by duplicating their content. This would artificially increase the number of hidden units and allow the model to learn more complex patterns" (P06.D). No respondents reported trying to derive classes or override class methods.

Viscosity

Q25 aimed to understand how is easy it was to make changes to code that uses API calls. Seven of 12 respondents mentioned it was easy and two mentioned it was very easy to make changes to API integration code (Q25)—"Easy, there was barely any code to write to implement the API." (P02.D); "Very easy. The interface is minimal and the actual parameters that one can change are few" (P12.D). Three respondents mentioned they did not need to refactor their code. The other two respondents described challenges around understanding the code in the context of refactoring it—"Easy as I wrote the code [...] When debugging issues though, I needed to check examples a lot to understand the described Test-Train-Run structure that I needed to implement. As in 'to train only once and not to run the model when testing or training'." (P01.D); "It was easy but needed a lot of understanding of the code." (P08.D). One participant referred to the growing amount of outputs as a difficulty for change—"As the amount of output parameters grew I found it sometimes difficult to keep track. Otherwise it was very easy" (P11.D).

Consistency

Q26 asked participants if they noticed API elements that offered similar functionality, and whether the differences between them were clear (Q26). Five of 11 respondents mentioned having noticed consistent method names across classes. Three of the aforementioned 5 found lack of clarity between certain API classes—e.g. "Model set, Regression and Classification. The difference between these objects was not clear. The implementation[s] were all very similar and it was not clear which one to use" (P02.D). There were also issues around the use of the different kinds of training data structures. The other two who noticed consistency of methods felt they understood the differences between them. For instance: "I like that there were a train, run functionalities in the code as this help me understand the models in similar way apart from the inner workings of course" (P01.D). The remaining respondents (6 of 11) did not notice such similarities; one participant did not respond.

Role-expressiveness

We asked participants if it was easy to read and understand code that uses the API (Q27), and whether it was easy to know which classes and methods to use (Q29). We obtained unanimous responses to both questions—"Everything was very easy to interpret." (P02.D); "Code is pretty self-explanatory and comments are concise enough" (P04.D) "Classes methods are efficiently named to understand what they are doing" (P08.D).

Domain correspondence

Questions about domain correspondence aimed to determine whether API classes and methods map easily onto the conceptual objects in the users' implementation.

We obtained unanimous positive responses about the ease of mapping the API code into developers' conceptual objects (Q30). Two respondents provided reasons that related the simplicity of the API interface and the IML workflow to the ease of mapping to domain and conceptual objects of their implementation (Q31)—"the simple user interface made prototyping very quick making building a conceptual idea very easy and simple." (P02.D); "I think because the training and the recognition phase is the same workflow, it's easy to come up with concepts that match both." (P07.D).

Participants seemed to have had a particular understanding of what was meant by the "mapping" of the API to an application domain (Q30); the majority of responses mentioned mapping API objects to classification or regression tasks, or to the IML workflow tasks. Most likely, participants have understood ML learning functions such as classification and regression, as enablers of functional mappings between domains (e.g. mapping gesture to sound, visuals, and discrete application events). This seems to be confirmed by the results of asking participants to provide examples of conceptual objects (Q31); only a few participants were able to refer to conceptual objects that did not overlap directly with ML domain concepts—“Once I had a clear idea how I wanted to activate certain functionality, this made the process easier for me.” (P01.D). “Because the API enable separation between recording” the data (i.e., training) and estimating the probabilities of category membership for an unknown example (recognition)” (P03.D).

Error-proneness

Questions about error-proneness aimed at eliciting the participants’ experiences encountering and recovering from errors in their use of the API.

Eight of 10 respondents reported that they had used the API incorrectly (Q33). Errors included: using an inconsistent number of data features between training data sets and test data sets (P05.D, P06.D, P09.D), using malformed data (P04.D), using labels inconsistently (P12.D) or malformed JSON (P05.D, P08.D), using large-size training datasets which caused a crash (P11.D), attempting to predict from a model in an untrained state (P02.D), and using a higher-abstraction level object as a primitive (P02.D). Many of these incidents were caused by limitations in input validation of API methods.

Four of these respondents indicated that the API did not provide sufficient help to identify misuse (Q34)—e.g, no error messages, some ‘undefined behaviour’ output. Participants reported having experienced crashes of the API-client application without any notification with the subsets XMM C++ (P04.D, P10.D, P11.D) and XMM JS (P09.D). One participant resorted to logging (P09.D) and contacted the API developers directly to find and resolve the issue.

Most respondents indicated they were able to find a way to correct their use of the API (35). For instance, where participants encountered errors due to lack of input validation, they adapted the library to implement validation (P05.D, P12.D). Other participants simply became more aware of problems and more careful (e.g. in structuring the training data, choosing the correct dimensionality of inputs and outputs, validating model state, etc).

Testability

Questions about testability aimed to determine the types of evaluation and assessment metrics that were adopted by participants as they used the API and concluded their implementation of integration code.

Most participants indicated having used subjective evaluation to assess the results of the trained models (9 of 12), with criteria such as correctness (3), cost (3), decision boundary characteristics (1). Several participants referred to other criteria such as expressivity (1) or more creative ways of evaluation—e.g. “No testing was done on the models, just eyeing up the output and judging it creatively whether it works or not for the desired output” (P01.D). One participant mentioned having used seam tests to assess training data. One participant did an objective accuracy evaluation of the models built with the API using unit tests with another ML library.

Seven of 11 participants found the API did not provide guidance on how to test the resulting application. The remaining respondents did not look for guidance for testing—e.g. “No, and we tested very informally since there’s no effective way to test more objectively” (P12.D).

5.8.3 Discussion

According to Clarke (2010), the CDs inspection can tell whether there are significant differences between what an API exposes and what a developer using the API expects. In this section, we use the results of applying the CDs questionnaire with RAPID-MIX API users to discuss design trade-offs with respect to developer experience and ML API usability. We also discuss the merits, challenges and limitations of the CDs for the assessment of ML API usability.

5.8.3.1 ML API design trade-offs in relation to learnability and understandability

Results indicate that the RAPID-MIX API qualifies as an ML API with a high or aggregate *abstraction level*. An ML API has direct *domain correspondence* if ML is considered its domain of correspondence. In the understanding of most users, the RAPID-MIX API entities map directly onto ML learning tasks. The high *abstraction level* is supported by its minimal surface area with a small number of classes, methods, and parameters. These elements have been subsumed into a simple conceptual model of high-level design tasks and basic data structures.

The high *abstraction level* appears to be consistent with the *learning style* of the RAPID-MIX API, which is more incremental and step-wise. Both novice and experienced developers reported an opportunistic learning approach (e.g., having hands-on exploration and progressing through code examples, exploring, changing or copying sample code to their projects). Arguably, given that ML learning tasks and the algorithms require extensive description from API providers and learning from the users, this indicates that the learning and assimilation of ML concepts was successful. ML APIs with these characteristics can provide ML -non-expert users with adequate scaffolding for a more satisfactory and forgiving learning experience.

However, more experienced developers reported to have complemented their learning strategies with a more systematic, top-down structured learning approach to the components and architecture of the API. More advanced developers and more technically complex scenarios may require the flexibility and control that a lower-level ML API with more primitives, more ML algorithms and more exposed parameters for finer-grained control can provide. We found that a few respondents, the more experienced developers or those who had specific implementation requirements (e.g., finer-grained control, strict end-user concerns within customer-facing projects) needed to go “beyond the interface" to inspect the API source code and learn more about underlying ML algorithms. In that exploration, a few of them found useful parameters that had not been exposed. This finding informed a subsequent re-design to expose the parameters.

In scenarios of exploration and intuition building about ML, ML APIs with surface-level *penetrability* may appear to provide everything that is required to enable successful use and integration with client application code. Nevertheless, surface-level ML APIs may allow 'black box' approaches in its application and use. We found that the RAPID-MIX API was no exception to this. As developers build up knowledge and understand the IML workflow, which ML tasks to apply, or the number of inputs and outputs to use in a ML pipeline, they may seek to push forward their understanding of a ML model behaviour. They may engage in a deeper exploration and experimentation to learn about the ML API intricate working details, such as the impact of choice of underlying ML algorithms and parameter change.

In the RAPID-MIX API, the overall *penetrability* is mostly sensitive to context and to implementation needs. We also found that different subsets of the RAPID-MIX API provided distinct levels of *penetrability.* There were cases of deeper exploration fraught with fragmented documentation, and unclear dependencies between API primitives and abstractions. This gives the RAPID-MIX API a core *consistency* rather than full *consistency.* These issues affect the perceived *consistency* of an ML API, and consequently, its learnability. For instance, some participants resorted to external resources to understand ML concepts and algorithms, which may be considered resorting to a top-down approach to learning ML.

Different areas of an ML API may have distinct levels of *role expressiveness*, which also affect its *consistency.* In most cases, the purpose of the RAPID-MIX integration code was correctly interpreted and matched user's expectations. Nevertheless, these issues prevented it from fully matching users' expectations which gives it a lower *role expressiveness* as an ML API. One opaque subset (i.e. RapidLib transpiled from C++ to asm.js) prevented one user from determining the underlying implementation. As mentioned before, other users found undocumented lower-level methods or lacked configuration settings. The transparency at the level of the ML algorithm—or ML explainability—is another layer that may entangle with the overall ML API *role expressiveness.* However, ML explainability is a current and significant research problem which is beyond the scope of this study.

5.8.3.2 ML API design trade-offs in relation to usability and applicability

An ML API with a high-level, easy-to-acquire conceptual model can cater well to the opportunistic approach and needs of ML non-expert developers. In the case of RAPID-MIX API, a simple conceptual model based on ML tasks and simple data structures with inputs and outputs, makes it suitable for simple implementations and rapid and pragmatic prototyping with IML. It also helps us to uncover and better understand usage and application of ML APIs by ML non-expert users.

ML APIs with a high *API elaboration* should not impede any kind of user from achieving their design goals. They should enable great flexibility to the more proficient end of the user spectrum, such as the implementation of custom behaviors, custom ML pipelines and parameterisation. Almost half of the participants reported using the RAPID-MIX API 'as-is' to meet their design goals. The other half required further API elaboration (e.g., more ML algorithms, more parameters, better error reporting). This suggests that for users with simple goals, the RAPID-MIX API was sufficient. Alternatively, it can suggest that, for more critical users, or, users with more sophisticated implementation goals, the API was not sufficient.

Arguably, the RAPID-MIX API exhibits a medium level of *API elaboration* as advanced users may use its extensible architecture to extend the API default capabilities with custom implementations. The few participants who extended the API default objects did so using adapter classes to extend the default objects and methods with validation, concurrency and error reporting. However, these users improved upon base limitations of the ML API. For a user, extending an ML API might defeat the whole purpose of using it in first place. Users who do not expect, or do not have the know-how to extend the existing functionality might it find problematic to do so. They may opt to use a different ML API or framework altogether, or resort to integrating independent ML algorithms.

Developers integrating an ML API in their client application code need to keep track of the information which enables them to work effectively (i.e., the *working framework*). Interestingly, half of the respondents did not mention ML algorithms as part of their *working framework*. This might reflect a trade-off with the *abstraction level* of the API; or alternatively, the adoption of specific ML API design assumptions (i.e., in the case of RAPID-MIX API, data and use cases on the foreground of users' attention and ML algorithms in the background). The lack of preponderance of the ML algorithm may be unsurprising if it reflects minimal ML requirements or a local *working framework* (i.e., ML API objects and methods, local variables) that suffices for simple implementations. However, the *working framework* may not be entirely or directly represented by the ML API or the scope of the ML API integration code—e.g., extrinsic elements such as the ML training data, or in a global or system-level working framework, client application and system configuration settings, external device data specifications, performance requirements, etc.

In a minimal test (e.g. hello world example, unit tests with a ML API) the *work-step unit* might be local and incremental. Despite the minimal surface area of a ML API, developers may have design requirements that scale the quantity of ML API integration code extensively. In these cases, an ML API can have a parallel *work-step unit*, where the steps to implement and achieve the full design goals are distributed throughout different scopes in the integration code. Given the interactive nature of the IML workflow, the ML API integration code will most likely scale up to comprise multiple and independent code blocks. This was the case with a few of the implementations with the RAPID-MIX API, e.g., asynchronous event handlers for collecting data and building an ML data set on the fly, for triggering evaluation of new data, or persistence to data repository.

ML API integration code may also require the instantiation of multiple auxiliary objects that interact together (e.g., GUI, data management, validation, concurrency), which make using and understanding more challenging.

Similarly, an ML API may support a *progressive evaluation* of the integration code at local level, functional chunk (that is, after the implementation of certain groups of tasks, such as setting data structures and training data set, or after the train and run methods), or parallel components (i.e., multiple and independent code blocks). The majority of respondents reported needing a fully functional pipeline and to experiment with IML workflow in order to check progress on the overall implementation task with the RAPID-MIX API. An ML API may support *progressive evaluation* at parallel components, as it requires a fully functional implementation and interaction between different ML API objects.

An ML API that presents the user with a small number of choices about how to accomplish design goals with minimal implementation differences between alternatives, can expose a minor and reversible level of *premature commitment*. The RAPID-MIX API also has a low level of *viscosity*, which allows users to easily make changes and refactor integration code. This is consistent with the notion that raising the *abstraction level* reduces *viscosity* (Green and Petre, 1996); low *viscosity* is also supported by the API small-surface area. Such ML API qualities invite a trial-and-error exploration and an opportunistic approach, and are supportive for ML-non-expert users.

The RAPID-MIX API situates at a medium level of *error-proneness*, given the reports about recurrent issues of misuse, error support and recoverability. These findings indicate opportunities and the direction for technical improvements, such as providing more robust validation of inputs, and better communication of error status through error messages.

Concerning testability, the RAPID-MIX API promotes more of a direct and informal evaluation using subjective criteria. This confirms its alignment with the IML approaches which the API is inspired on. In any case, most developers seem to be unaware of different methods or evaluation alternatives, and seem to find the concept difficult to articulate. Also noted was the lack of guidance about evaluation alternatives, which seems to require specific ways to be successfully transmitted, such as with richer media.

5.9 Limitations, threats to validity, and mitigation of potential bias

In the UCD actions for lightweight evaluation of the RAPID-MIX API, I had to adopt a "lightweight" approach to deal with the intensive schedule, lack of budget (both time and money), and lack of collaborators willing to help on all the required stages (preparation, deployment, analysis and validation). As such, these studies are inherently limited and biased. Strategies that I used to mitigate bias included triangulation of different primary data sources (observation notes, videos, code), conditioning the data, the data analysis and findings to expose them to the consortium. I also invited other researchers to perform “cut-throughs” and “deep dives” on the data. This initiative did not have much success, however, as the tasks involved going through such an extensive amount of data are extremely daunting and there were not sufficient motivators to do so.

In the CDs study, we adopted a user-centred approach to the design and evaluation of an ML API, and explored the application of the CDs as a broad-brush usability assessment technique. We were able to apply the rich vocabulary of the CDs framework to the analysis and discussion of an ML API design, informed by the perspectives of a limited and biased sample of developers using the RAPID-MIX API in the real world. We were able to relate subjective CDs of analysis and their interactions to ML API design tradeoffs, to broad elements of ML API developer experience (i.e. learnability, understandability, applicability and usage).

The application of the CDs framework to ML API design and assessment yielded different benefits. Mainly, the CDs framework opens up interesting perspectives of analysis which support a rich and deep discussion about ML API design. This is an important outcome to designers of ML APIs who may be looking for methods to quickly understand and obtain distinct perspectives into their ML API designs. The findings obtained through the CDs cannot lead to full validation of the usability of an ML API design (Dagit et al., 2006), particularly with such a limited sample of users. However, they can lead to new insights which may trigger new design iterations (e.g., refactoring abstractions by exposing hidden dependencies between inputs, outputs and ML models; improving the abstraction level for prototyping with IML by exposing more ML parameters; improving the documentation for learnability, etc). As a method of inquiry and broad-brush analysis, the CDs framework can be, therefore a useful and pragmatic resource for ML API designers.

Mostly, we found challenges in the general application of the CDs, rather than in its specific application to ML APIs. We found interpretation problems with less experienced developers. In one instance, one respondent and recent CS graduate complained about never having thought about APIs and ML in those terms and finding difficulties with the redundancy of questions. This accords with the literature that shows that experienced developers can have an easier and more intuitive grasp of the subjective concepts of the CDs (Green and Petre, 1996; Petre, 2006). We also found that to communicate the CDs vocabulary to designers to use in their discussions, is not straightforward, particularly in time-constrained situations. In a first instance, the CDs vocabulary just appears to be more jargon that needs to be acquired. The degree of subjectivity and interdependence of the CDs makes it even more challenging. Furthermore, as we found in our pilot study, the length and redundancy of the CDs questionnaire, which we believe is purposefully counteracting issues around subjectivity, creates typical problems of frustration, which may undermine the overall assessment effort.

The main limitations which we found in the application of the CDs to the assessment of ML APIs are in the analysis and interpretation of the results. First of all, it is difficult to establish a scale for each dimension of analysis. Scales rely on the assumption that participants have the same or similar perceptions of scale responses, when in fact they are subjective. Also related, is the difficulty to rate an ML API for each dimension of analysis. We found these issues particularly meaningful in our attempts to employ the scales used by Clarke (2010) for analysis, discussion and when trying communicate our reasoning about individual dimensions.

Other limitations intrinsic to CDs are related to interactions with other artefacts, such as the text-editor or programming environment, in which the ML API integration code is programmed, or the documentation media. Other broader aspects where CDs are limited are time and context and how these affect CDs interactions. For instance, *penetrability* and *learning style* are CDs which are likely to change over time and context. These issues have been identified in previous research (Petre, 2006). Finally, findings obtained through CDs can be challenged in terms of validity and reliability, as with other questionnaires and surveys (Adams and Cox, 2008). The reliance on participant opinions must be considered, as well as the impossibility of measuring with strict reliability the objectivity of the participants' responses.

5.10 Summary of second Action Research cycle insights

The RAPID-MIX API can be understood as a bundle of software components, services and resources—an ensemble that emerged from the organisational context of RAPID-MIX through a collaborative effort in design and development. The RAPID-MIX consortium employed concurrent prototyping around a set of scenarios and use cases of interest to define the RAPID-MIX API design guidelines. RAPID-MIX consortium stakeholders grouped and engaged in parallel tracks of concurrent prototyping to understand possibilities and actively shape the design directions for RAPID-MIX technologies—which illustrate the *reciprocal shaping* (Sein et al., 2011) between technology and the organisational context in RAPID-MIX. This was an important step in the RAPID-MIX API design process which led to the identification of core ideas and features to implement. The parallel prototyping and evaluation activities led by the RAPID-MIX consortium also contributed with shared insights towards the distillation of RAPID-MIX API design guidelines.

As illustrated by the concurrent prototyping activities and key instances of personal communication from RAPID-MIX stakeholders, as participants of the design process strove to reach consensus on the main characteristics and features of highest value of the RAPID-MIX API, they were learning from each other—an instance of the *mutual influential roles* (Sein et al., 2011) at stake in RAPID-MIX and in the development of the RAPID-MIX API. The emerging design directions contributed to innovative and unique features that aimed to fill a gap among the available ML libraries and toolkits, with special concerns about usability, user-friendliness, breadth of reach, accessibility, and affordances for rapid prototyping.

The real-time IML and co-adaptive approach sets ground for new assumptions based on functional value, such as providing more effectiveness than other techniques, for certain situations, tasks and users. For the RAPID-MIX API, this value has been identified in building functional interactive systems more effectively using rapid prototyping. This value is an important contribution to design and incremental innovation processes, or the development of intuition applying ML more effectively—i.e. when compared with approaches that use off-line batch training, slow training algorithms, large data sets, etc.

The RAPID-MIX API design guidelines focused, therefore, on utility, usability and architecture of the infrastructural software. They prescribed a multi-layered and modular architecture with interoperable components. Based on the results of the previous UCD actions, they also prescribed support for end-user diversity, focusing most notably on developers who qualified as ML-non-expert users, with a wide range of programming skills, from novice to advanced, with specific domain expertise.

Concerning the utility of the RAPID-MIX API, the wide range of potential use cases that was perceived from the initial set of broad-scoped and general purpose RAPID-MIX technologies was narrowed down to i) building mappings for digital music instruments, and ii) mobile applications and game controllers using gestural, audio and other sensor-based and multimodal data analysis. Most importantly, the RAPID-MIX API was designed as infrastructural software for rapid prototyping and exploration of end-to-end, real-time, multimodal, machine learning pipelines, for leveraging IML workflow across a broad range of applications, programming environments and deployment targets.

The concentration of prototypes, proofs-of-concept and RAPID-MIX background technology integration in the dimensions *Interactive Machine Learning* of *Component Technologies* as well as in the use cases *Interactive Machine Learning and Data-driven Custom Interaction* (Table 5.4) illustrated the design focus which occurred in this dimension of the design space. RAPID-MIX API implementation and documentation provided support for and promoted the integration of the IML workflow in client applications. This meant supporting interaction between end user and ML algorithms, end-user curation and modification of training examples, and a direct, intuitive, and subjective evaluation of the results of applying the ML model.

In order to fulfil these requirements, the RAPID-MIX API was built as a bundle of software components distributed between different codebases and layers in C++ and Javascript. The unification of these components was achieved with a high-level unifying interface in C++, transpilation to Javascript, and implementation of Javascript clients of server-side ML native code. The RAPID-MIX API also bundled and centralised access to an ecosystem of resources that included documentation, demonstrators and infrastructural services. For instance, the RAPID-MIX API website with interactive tutorials, CodeCircle as a online collaborative programming environment, and Repovizz 2 with a service for multimodal data repository with a RESTful client API.

The interventions that I designed and deployed in collaboration with other RAPID-MIX stakeholders illustrate how the *authentic and concurrent evaluation* (Sein et al., 2011) of RAPID-MIX API yielded fruitful outcomes. Two UCD actions for lightweight formative evaluation of different subsets of RAPID-MIX API and target user groups enabled a better understanding of RAPID-

MIX API users with regards to their needs, goals, and expectations, and to the challenges they found using RAPID-MIX API. A study with the cognitive dimensions framework provided us with a broad-brush and structured evaluation of the components of our toolkit, concerning its design trade-offs and usability.

We found that most of the RAPID-MIX API users, in particular those at the low end of the spectrum of technical expertise, expressed needs related to *ML conceptual knowledge*. These needs were mostly framed in terms of learning resources that could more effectively support the acquisition of ML concepts. These included resources that could i) provide the big picture of ML, and ii) express the relation between ML concepts, and iii) provide an incremental and engaging approach to learning, including resources that could further their learning experience about the different algorithms underlying RAPID-MIX API.

Additionally, RAPID-MIX API users expressed needs for learning resources related to the *application of ML*. For instance, RAPID-MIX API users expressed needs for code examples bridging ML concepts to an implementation with the RAPID-MIX API. Particularly, RAPID-MIX users required code examples written specifically for learning and understanding—e.g. with segregation of boilerplate code, with consistent naming practices of program variables across examples, and without excessive sophistication or performance optimisations. Users of the RAPID-MIX API also showed a need for having code examples with design patterns for the most frequent use cases. In particular, developers working with the RAPID-MIX API showed that examples built as ML building blocks could be useful for simultaneously learning about ML and using it effectively through rapid integration in users' projects.

We found commonalities and patterns across users more directly related to the hands-on application and usage of the RAPID-MIX API:

- Our ML developer tools were considered easy to use and providing fast results.
- The lack of feedback in the training stage of the IML workflow to be a critical usability problem. This problem posed important implications, including undermining the adoption of the RAPID-MIX API for future use.
- There were users making competent use of the RAPID-MIX API, who showed difficulties in understanding data, and how to work with it, even before using machine learning. There were instances of application of the IML workflow correctly without understanding what features were and how they were calculated, or, how filters such as the moving average worked, what it did to the original data, or the overall impact of these design elements in certain design moves. This caused users to make operational mistakes that they did not understand and hit barriers they could not overcome on their own—e.g. training a ML model with raw data and testing it with filtered data.
- Many users of the RAPID-MIX API designed applications where they employed ML using

"black box" testing approach. They used ML algorithms without understanding the their assumptions and their inner workings.

- Most users employing the RAPID-MIX API to build applications subjectively evaluated the outcomes of IML workflow and considered them sufficient for their goals.
- A few users employed the RAPID-MIX API to build developer tools with a higher abstraction level—e.g. software infrastructure for gesture recognition in game design, JUCE framework component for employing IML in audio software development, a port of the RAPID-MIX API for Bela, sand boxes for Unity, Max/MSP, React. Users embodied their domain-specific design knowledge in these tools which included infrastructural software for others to use, and provided an incremental innovation step to scenarios they were knowledgeable of. There were end-user application such as sandbox applications for probing variations of specific combinations of inputs-features-algorithms-parameters. There were also "second-order" infrastructural software included libraries that wrapped around the RAPID-MIX API, to adapt its core functionalities to a specific development platform or framework, or to build domain-specific metaphors (e.g. left-hand detector for Leap Motion data).

We were able to do a more clear characterisation of RAPID-MIX API user archetypes through the definition of a set of design personae, which includes:

- The experienced audio software developer who systematically builds low-level and highly performant code and is looking at ML developer tools for rapid prototyping and improving his design and development processes.
- The web designer and developer who engages with OSS practices in web development tooling ecosystems and web innovation communities, and has a pragmatic approach to software development. She wants to quickly grasp the benefits of IML and develop intuition for using ML, and show it through the development of proofs-of-concept on the web.
- The creative computing student who has a broad but relatively superficial understanding of different development technologies. He approaches software development opportunically and wants to get an edge in rapid prototyping with ML developer tools, for hackathons, to show and tell his skills and earn reputation.
- The maker and tinkerer who dreams of entrepreneurship and takes an opportunistic approach to building prototypes with embedded hardware and real-time sensor data, where ML can provide an edge.
- The music composer and audiovisual artist who has developed an intuitive understanding of ML and IML with high-level tools such as Wekinator. Although this user might not be able to code a ML application with textual languages, he can use visual components and create diagrammatic patches with data flows, in high-level and visual programming environments

> such as Max/MSP, Pd, Isadora, etc. He is looking for more flexibility and to engage in partnerships with developers or community efforts that can produce components and higher-level tools more suitable for his needs.

The depiction of design trade-offs that CD inspection provides can inform about the usability of the RAPID-MIX API. Results indicate that the RAPID-MIX API is usable and well suited for the needs of creative developers. It seems to cater particularly well to beginners and ML-non-expert users in general. The API design features an effective support for a low entry barrier and smooth learning curve to ML, through incremental learning approaches.

The direct correspondence of the API to a conceptual model that focuses on supervised ML learning tasks also supports effective learning and understanding. The structure and entities of the API support usage with minimal amount of code and context, trial-and-error exploration, easy refactoring and easy adaptation to custom user needs. This facilitates opportunistic development approaches, that are driven by design and rapid experimentation, and prone to happen in the context of creative development.

RAPID-MIX exhibits a trade-off between the high abstraction level and a surface level of penetrability, which makes it more challenging for users to learn the intricate details that more technically complex implementations might require. There is also a trade-off between error support and performance. These trade-offs may be limiting for users who require a deeper understanding of the underlying technology and more defensive approaches to development, such as seasoned developers opting for a systematic development approach.

The cognitive dimensions confirmed that the RAPID-MIX API is a usable and useful API for rapid prototyping the IML workflow in proofs-of-concept and end-user applications, as well as in the commercial development of creative technology and viable multimedia products. The study has shown that the RAPID-MIX API is usable for:

- creative developers—people with different software development skills, with beginner to intermediate ML experience, developing creative technology;
- different scenarios—e.g. personal use, proofs-of concept and commercial applications in creative technology;
- specific use cases—building mobile audio applications, digital musical instruments, game controllers, audiovisual installations and performances, etc.

In the process of *authentic and concurrent evaluation* (Sein et al., 2011), we identified usability problems mostly related to documentation fragmentation, to feedback on ML training, error support and recoverability, and lack of evaluation guidance. We found that the most important directions for improvement concern the provision of more effective documentation for ML conceptual knowledge and examples of application of ML code, and support for understanding intricate working details. These findings helped to provide recommendations for the improvement of the design of RAPID-MIX API and its documentation resources. These UCD actions also helped to uncover some of the specific challenges of applying UCD in a multi-institutional innovation action such as RAPID-MIX and with a developer toolkit for machine learning.

It is therefore absolutely obvious that a simple notation that just hides the complexity in the definitions of symbols is not real simplicity. It is just a trick. [...] When you unwrap the whole thing, you get back where you were before.
— Richard Feynman

Chapter 6

Discussion: User-Centred Design of Machine Learning Developer Tools

This chapter re-examines one of the main research questions and discusses the contributions for research around user-centred design of ML APIs, based on the approach used with the RAPID-MIX API.

6.1 Introduction

In RAPID-MIX, a user-centred approach has been critical to ensure that our ML developer tools were usable and useful for creative ML-non-expert developers. The needs of developers using ML were not well-understood (Patel et al., 2010) and the needs of people employing technology used in design and other creative practices have often been considered distinct from people engaging in activities with more well-defined outcomes (Cherry and Latulipe, 2014). We have thus needed to carefully craft a user-centred approach which was appropriate to the design of the RAPID-MIX API within the additional constraints of a complex project with multiple academic and industry partners.

6.2 Understanding the multiple contexts of ML infrastructural software users

Olsen (2007) proposed the concept of *situation, tasks and user* (STU) *contexts* for establishing the setting that he considered necessary for assessing the quality of interactive systems innovation.

Most relevantly, Olsen used the case of user-interface toolkits to illustrate how the concept of STU contexts can be useful to clarify ambiguities about the context of use, users and tasks. In STU contexts related to toolkits, users are developers with the task of producing applications and interfaces for another STU context, i.e. for specific end users to fulfil end tasks in an end situation. An STU context is therefore a particularly useful concept in this discussion, to help clarify ambiguities between the RAPID-MIX API and its client applications, concerning tasks and contexts of use and how they may or may not be different. STU contexts also help to clarify between RAPID-MIX API users, proxy users of the IML workflow—i.e. developers, designers, or people who employ their judgement concerning end users in user requirement elicitation for design activities of ML applications (Preece et al., 2015)—and end users of client applications of the RAPID-MIX API.

Under an STU context perspective, RAPID-MIX API users can be end-user developers, building for themselves, or proxy users, building for both end users and other proxy users. In the primary STU context, the user mostly focuses on software engineering and programming, i.e. interacting with data structures and objects of the RAPID-MIX API, programming ML pipelines for building high-level interfaces. In secondary STU contexts, users can be end-user developers or end users, employing an IML workflow using a high-level interface for iteratively creating, curating, and modifying training datasets, and building and steering the behaviour of ML models. They can also be developers creating higher-level interfaces and abstractions of the RAPID-MIX API.

Primarily, the RAPID-MIX API was useful to developers in a primary STU context, in which they were users of ML tools for their development tasks. The primary STU context received focus in the RAPID-MIX API design, which targeted as users developers creating intelligent systems for themselves—e.g. end-user developers (Lieberman et al., 2006)—or for end users. Users from this STU context, as developers, interact with ML algorithms primarily through APIs such as RAPID-MIX API; they interact with data through data structures, program ML pipelines and build interfaces, including GUIs, to expose ML control and feedback functionalities, and higher-level APIs, for users of secondary STU contexts. Section 5.7.5.2 characterised in detail the different typologies of RAPID-MIX API users that fitted the profile of primary STU context using design personae.

The term *end-user developer* (Lieberman et al., 2006) can be applied to a RAPID-MIX API user when, as an original developer of an end-user application, she is using it for herself—e.g. for testing the application or applying it realistically and intentionally to an end-user task. However, she is a user in a different situation and performing a different task when she is interacting with her newly developed system or interface using an IML workflow. There is a secondary STU context implicit here, which can refer to an end-user developer—the same developer who initially built an application client of ML infrastructural software—when she uses her newly developed system for herself and for executing an IML workflow, for iteratively creating, curating, and modifying training datasets to build ML models and steering their behaviour. In this case of an end-user developer, primary and secondary STU contexts share the same user in different tasks and situations (developing with RAPID-MIX API, using IML interfaces built with the RAPID-MIX API for an end-user task).

The term *end-user developer* can also be applied to a “real” end user in a secondary STU context. In this case, end users can be domain experts using domain-specific IML systems built with RAPID-MIX API, with high-level interfaces that support the IML workflow. These users build ML models for a domain-specific problem or concept and steer its behaviour by iteratively creating, curating, and modifying training datasets. In this secondary STU context, the end user leverages on the adaptive nature of ML and on the ability that an IML interface provides her—i.e. “programming” by demonstration, configuration, or customisation of the end-user application (Lieberman et al., 2006). In both cases, they are end users of the real-time IML workflow, in which they iteratively modify training data or learning parameters, to “steer” an ML model towards the desired behaviour. RAPID-MIX API would not only be useful to developers as users of ML algorithms and as “proxy-users”—i.e. producing real-time IML workflow interfaces for other STU contexts—but also for pushing the advantages of the IML approach to the end user.

One secondary STU context which may appear to be subtler at first sight, but revealed one of the most interesting findings of this research through the RAPID-MIX API design, refers to end users who are developers and second-level proxy users of the RAPID-MIX API. These users create higher-level programming interfaces (e.g. wrappers and components) of the RAPID-MIX API for other proxy users. These users take design decisions about API and component development with reference to other developers, or to other domain- or problem-specific end contexts. They produce their own toolkits, for instance, by refining the abstraction level of the API, or providing specific default parameter configurations; or, exposing or hiding ML building functionalities and domain-specific metaphors based on their designer knowledge of a specific STU context—producing APIs or components that end users use to apply IML in their specific end situation. These users can be understood as a particular case of *Lead Users* (von Hippel, 1986), who not only make use of a ML toolkit to customise or create their product, but also appropriate of the toolkit to provide a new version adapted on their specific design needs and understanding of the needs of a specific domain of applications.

We found instances of this pattern with participants in both UCD actions and with RAPID-MIX API users who participated in the cognitive dimensions study. For example, ROLI hired a developer to create the JUCE ML module which was deployed in the JUCE ML hackathon. This module consisted of a wrapper around the RAPID-MIX API C++ subset for JUCE users and developers to employ ML in their applications. As previously described in Section 5.7.1, most participants integrated the RAPID-MIX JUCE ML module into their applications. Interestingly, one participant of the JUCE ML hackathon “peeled off” the JUCE ML wrapper code from the RAPID-MIX API, having done so autonomously and to directly adapt the RAPID-MIX API to his own needs. He adopted the machine learning functionality into the programming environment of the Bela embedded hardware platform, and adapted the RAPID-MIX API classification and regression class data types to make IO work in the Beaglebone Xenomai Linux environment (the operating system that runs on the embedded hardware that hosts the Bela board).

We observed other instances of this pattern with participants of eNTERFACE'17. Participants had more time at eNTERFACE'17 and three of them built their projects and put their own toolkits together. These 'second-order' proto toolkits—i.e. early versions of toolkits which consisted of different versions of the RAPID-MIX API that have been adapted according to users' needs for others to use—varied in shape and functionality. For instance, *Gesture Server* was a toolkit that one participant built out of a server-side application with the XMM C++ subset of RAPID-MIX API. He abstracted all the machine learning and terminology, an instance of temporal classification with Hierarchical Markov Models, into a domain-specific remote service with an API which provides mobile phone gesture recognition. End user of this service could be perfectly unaware they would be dealing with ML unless they were customising the gesture by train the models. Another participant of eNTERFACE'17 engaged in a broad exploration to build applications that mapped different configurations of sensors, ML algorithms and client application. Simultaneously, she was curating a toolkit to deliver in her creative computing classes to her students.

The work of three of the cognitive dimensions questionnaire respondents also supported this pattern. They also employed the RAPID-MIX API for their own needs to build a specific application and in doing so also extracted their own version of toolkit using the RAPID-MIX API. One of them built a Max/MSP external, a reusable module in the Max/MSP environment for employing the IML workflow in a data flow programming environment. He used this component in an experimental Max/MSP standalone application, a tool for helping musical therapists working with disabled children using different configurations of sensors, machine learning tasks and musical processes wrapped in specific metaphors. The other participant also used the RAPID-MIX API JS in a data flow programming environment, but he built his own from scratch in React, a web development framework. This application enabled the user to customise real-time data pipelines from multiple data sources classification, regression, for specific application behaviours.

An STU context characterisation helps to set a clearer picture of the different STU contexts for better understanding RAPID-MIX API users and discussing the innovation, usefulness and importance of the RAPID-MIX API design. An STU context characterisation can provide useful framing for a more general and holistic understanding about the implications of ML developer tools to the users and tasks of the different STU contexts related to those tools and their client applications. The advantages of the real-time IML approach have strong implications concerning features that RAPID-MIX API client applications provide to secondary STU contexts. The benefits of the RAPID-MIX API—i.e. supporting the development of real-time IML systems, facilitating rapid prototyping and certain programming tasks, encouraging users with a lack of ML proficiency to learn and use ML algorithms, and empowering them to "program" by demonstration and customise interactive systems.

6.3 Understanding ML infrastructural software users and their developer experience

As with general API design (Clarke, 2011; Myers and Stylos, 2016), a user-centred approach informed and validated design decisions of our ML developer tools with usability data. We employed this approach in the systematic gathering of data and information about the users of our ML development tools, concerning their goals, needs, values, and expectations (Chapter 4). As the RAPID-MIX API was developed, we employed our user-centred approach in its validation and evaluation with users (Chapter 5, Sections 5.7.1 and 5.8). We looked for usability issues, conceptual knowledge gaps, challenges and mismatched expectations. The findings from these studies informed and were used to refine the design of the RAPID-MIX API.

The information we gathered showed that, in a first instance, the primary goals of creative developers with little or no proficiency in ML are to rapidly acquire literacy by developing an intuition and basic understanding of ML and to be able to apply ML tools creatively in their practice. As discussed in Section 5.7.5.1, ML -non-expert developers were mainly motivated by curiosity, triggered by an opportunity to explore ML in practice by applying it to a pet project, by enticing examples of ML applications, or by the buzz around AI. ML -non-expert developers are motivated to enhance their development skills and add ML tools to their arsenal of development techniques and problem-solving competencies.

As with other developer tools, developers need to acquire a certain level of conceptual knowledge which, as reviewed in Section 2.1.1, may include a subset of ML terminology, workflows, learning strategies and tasks, and algorithmic implementations and parameterization. However, even a minimal and representative subset of what is prescribed as 'folk knowledge' by the applied ML community (Domingos, 2012) can become overwhelming for novice users. In our studies with a diverse set of ML-non-expert users, we observed that the notion of ML 'folk knowledge' may vary for different levels of experience and knowledge in ML learning. Given the breadth, depth, and interdisciplinary nature of the ML knowledge domain, it is necessary to adjust this notion of 'folk knowledge', when the goal is to promote user engagement, learning and adoption of ML developer tools. The suggested level of ML knowledge to be attained should be spread through progressive levels of complexity, and using learning devices that can support a gradual learning experience while carefully managing cognitive and information overload, particularly with beginners.

In our UCD actions, we found different user needs. Some of these needs were expressed directly by the users, while others were derived from observation and data analysis. For instance, in Section 5.7.5.1, we discussed how participants of the UCD actions, starting their ML learning experience from the ground up requested informational elements that could provide the big picture of ML concepts and practical, observable and immediate results. They requested information that highlighted the relations between the ML conceptual knowledge and terminology and how ML developers tools implemented them. Users suggested and requested artefacts that would concentrate this broad knowledge in useful ways such as mind maps, infographics, or cheatsheets. Users also requested documentation structured with an "incremental approach", "recipes", or short stories.

Another interesting finding related to ML conceptual knowledge was the different ways the IML workflow was communicated and how quickly participants would get the gist of it. This finding was based on observation of participants and would benefit from further empirical and more formal research. We found that live or rich media-based demonstrations of the IML workflow appeared to be remarkably effective, more so than when conveyed with static sources of descriptive knowledge, such as text—e.g. documentation, code, or code comments. This might be explained by the procedural nature with which we communicated the IML workflow, inspired by the work of Rebecca Fiebrink with the Wekinator, using live demonstrations and richer media, such as video and animations, which were very effective at conveying the workflow and yielding a faster introduction.

Developer tools with different design trade-offs in terms of abstraction level and penetrability (Section 5.8.3) provide a different scaffolding to the learning experience of users with different levels of experience in ML. The RAPID-MIX API has shown some success at providing an introductory ML experience, getting beginners to advance in their practice and support for rapid prototyping. Features such as a simple conceptual model for organising ML algorithms within a framework of design tasks and simple types of data, standardisation of high-level programming interfaces for ML algorithms to simple tasks (i.e. record, train and run) can be effective in the achievement of learning outcomes, in supporting immersion in the task, and practical results—mostly simple end-user applications that are functional and satisfactory. ML developer tools that aim to additionally support more advanced levels of ML experience and user needs—i.e. both the low threshold and high ceiling—face the design challenge of providing different levels of abstraction, coherent mechanisms for transitioning between these levels, a high level of transparency and penetrability.

These findings motivated the creation of learning guides with embedded real-time interactive examples on the RAPID-MIX API website. These learning guides provided immediate immersion and supported a hands-on approach to a curated set of foundational ML concepts. The "Getting started" section showed how to implement IML using an incremental approach through contrasting examples—e.g. pre-defined vs dynamic training data set, regression vs classification—leading up to concrete, specific use cases—e.g. implementing mappings between real-time sensor data and synthesizer parameters. Such examples can illustrate how to start from the most simple ML examples, with a small pre-defined data set with two ML classes, and evolve to more complex examples using real-time data streams, through to building up more compelling use cases grounded on real world scenarios and real sensors such as the LeapMotion or the Myo.

Later design moves focused, therefore, on improving the RAPID-MIX API documentation, on making it interactive, using end-to-end ML pipelines and the IML workflow. This fulfilled the critical need of ML-non-expert developers for practical know-how, for employing the ML tools in practice. They need documentation with examples, tutorials, complementary knowledge about supporting tools (i.e. secondary tools that support ML developer tools such as IDEs, code repositories and versioning systems), and potential community support for finding mentoring, for troubleshooting and difficulties, and short-cutting efforts towards their specific goals. Such elements comprise the baseline of what developer tools should offer in general. Typically, developers expect the same from ML APIs, given that developers are usually granted access to these supporting resources as part of tooling ecosystems for other middleware and APIs.

Although ML-non-expert developers say they need tools that support open-ended exploration (Section 4.4.3), we observed that they also need a strong proposition in terms of the conceptual scaffolding. This is the case, particularly, if they are starting with ML from the ground up, or charting new grounds in their knowledge on a specific avenue of ML. As ML-non-expert developers develop basic knowledge about ML, they may become interested in attaining a deeper understanding and they require more flexibility. For instance, for some of the developers we engaged with, the conceptual part about algorithms was considered both important and lacking in the early documentation. Users asked for more insights about how the algorithms worked and looked for succinct explanations, links to further resources, including external third-party resources, such as videos, papers, blogs, etc.

Curation of external learning resources plays an important role when providing for the ML learning process, and therefore, also in the design of documentation for new ML developer tools. But, most importantly, we found that as developers advance in their exploration, depending on their specific needs, they seek to go beyond the ‘black box’ towards understanding its core. For instance, developers needing a certain degree of accuracy (e.g. as per design requirements of their projects) want to quickly explore the ML algorithm design space, by probing it and tweaking its parameters on small but representative data sets. They want to understand the limitations of the algorithms against their use case. This was confirmed both with advanced developers in UCD actions and in the cognitive dimensions questionnaire.

6.4 Informing the design of infrastructural software for rapid prototyping with IML

The information obtained through a user-centred approach supported the gradual development of more principled and specific user models of ML-non-expert developers and of their interaction with ML developer tools. These models can be used to inform design decisions or improvements to the design of ML developer tools. User models can be created and established as representations of knowledge about users, which can then be shared between system designers and other stakeholders.

There are different types of user models that can be created for this purpose, ranging from implicit (only existing in the user-centered designer’s mind) to explicit, and from simple to more complex (Ritter et al., 2014). They can include, for instance, user profiles, design personae (Cooper et al., 2007), implicit descriptive models (Ritter et al., 2014), through to formal models, such as cognitive architectures and information processing models such as the Model Human Processor (Card et al., 1990). While complex and formal models are more typical in formal approaches to usability engineering and human factors (Ritter et al., 2014), the user-centred and generally pragmatic approach in RAPID-MIX has mostly employed implicit user models, which emerged from the frequent contact with end users via UCD actions, and design personae, which I progressively developed to formalise the knowledge about RAPID-MIX API users.

As recommended by the literature (Holtzblatt et al., 2004; Ritter et al., 2014), UCD was applied early and throughout the lifespan of RAPID-MIX, with consortium partners, potential end users, and real users of the RAPID-MIX API, the design artefact and unit of analysis of this thesis. Through the deployment of UCD actions in RAPID-MIX, we were able to evolve from an initial basic user specification, based on type of technology and project role (Table 4.1), to an affinity map of speculative end-users (Appendix C.5), to a more scoped and composite profile (Section 4.6), through to a more principled user model described by design personae (Section 5.7.5.2).

In the preliminary, up-front design stage (Chapter 4), a UCD approach motivated a better understanding of the needs and goals of the network of stakeholders, including RAPID-MIX consortium stakeholders and potential RAPID-MIX technology end users. In turn, this motivated the definition of a collaborative research strategy for RAPID-MIX consortium stakeholders to engage end users with interventions to better understand their needs and goals, and the design and formalisation of the UCD Actions methods framework. The UCD Actions framework was designed as a pragmatic research tool to guide RAPID-MIX consortium stakeholders in designing and deploying UCD actions with end users in plausible contexts, such as in the concurrent prototyping and evaluation activities.

The usability data that we obtained from UCD actions with potential end users at this preliminary stage enabled us to establish basic user profiles with an initial characterisation of background, technical skills, needs, expectations, and challenges, as well as perceived advantages and limitations of candidate RAPID-MIX technologies. The initial user profiles that we have formulated included developers (of creative applications for web, mobile, audio and games, academic proofs-of-concept) and designers (interaction, audio and visuals, educational content, interactive performances, collective experiences). These user profiles considered common traits of having little or no experience in ML, and interest or background in computational creativity applied to design and production of artefacts for personal use, and for cultural and commercial exploitation.

Contrastingly, our later UCD actions (Chapter 5) enabled us to validate our design decisions and refine our design. For instance, we found that more experienced developers with more sophisticated development skills expected flexibility and the performance of low-level development tools. These findings provided not only useful information but also a dialectical tension, as it implied more complexity in the design of RAPID-MIX for supporting flexibility and striking a satisfactory balance between ease of use and expressive power.

Designing the RAPID-MIX API as a toolkit with the adequate level of flexibility implied providing a low entry barrier—to facilitate the adoption by less experienced developers—'wide walls' for exploration, and a high ceiling, with an extensible architecture that could provide for the specific needs of more experienced developers. This led to establishing principles for designing tools with a simple conceptual model, which organised a selection of machine learning algorithms in a modular and extensible architecture, simplified the workflows to small number of tasks or operations, and narrowed the set of target use cases.

Our research findings also contributed with information about the space of compelling applications for RAPID-MIX technologies. We found a general perception about background RAPID-MIX technologies concerning their broad scope, general purpose, and complexity, both from RAPID-MIX consortium stakeholders and potential end users. We also found consensus between stakeholders and potential end users regarding general design features, such as supporting cross-platform development and multi-target deployment. We found categories of prototypes that indicated more focused areas and products of interest, which we used to refine and narrow down the number of use cases of the RAPID-MIX API to present to the users.

We also found domains of application, such as rapid prototyping and digital instrument building, and formats to deliver our tools, such as workshops for ML teaching, which were adequate for our target users. The perceived importance of a community by RAPID-MIX API end users also gave an impulse to the RAPID-MIX API design to leverage on the affordances of the social web. As such, RAPID-MIX developed CodeCircle, an online platform not only to host community interactions, but also to provide direct access and usage of integrated ML developer tools for rapid prototyping, and to host creative interactions with ML developer tools. This contributed to lowering the entry barrier and supporting a more immediate engagement with our ML development tools.

The Cognitive Dimensions framework allowed us to apply a rich vocabulary in the discussion of API design, informed by the perspective of developers using the RAPID-MIX API. The CD framework opened up interesting perspectives of analysis and supported a rich and deep discussion about ML API design. However, the CD framework is not an easy method to apply. The dimensions appear to have a certain degree of subjectivity and interdependency. This makes it challenging to proceed in analysis and interpretation, as well as in rating the artefact for each dimension of analysis. Nevertheless, it contributed a deeper understanding about how ML API design trade-offs are experienced differently by the users for whom the API is designed.

The set of target user personae revealed heterogeneous technical skills in software development and needs. We considered both users with limited programming experience and with professional-level development skills (mostly, in audio software). We found that for many potential users, working with low-level APIs presented a low appeal and, instead, they would prefer new ML developer tools that could be integrated in the developer environments they already used. In fact, this was one important finding that was recurrent throughout action research cycles and UCD actions; the integration of RAPID-MIX technologies with development environments with which users were already familiar and which they used in their projects. RAPID-MIX consortium stakeholders also suggested that this pipeline approach could be integrated in different prototyping and programming environments. The decisions that led to designing, building and providing user-centred ML infrastructural software for integration in these environments were congruent with these findings. They were an adequate approach for broadening and accelerating innovation with ML developer tools.

6.5 Conclusions

UCD has most often been applied to the design of physical artefacts and graphical user interfaces. However, the application of UCD to technologies, such as middleware or ML is not as straightforward. In such cases, it might be challenging to verify success or recognise the benefits of such an approach.

The practical implementation of UCD within the large RAPID-MIX scale was of a complex and challenging nature. Our experiences accord with Norman's (2013) remarks about the compromise between the UCD philosophy ideal and its practical implementation problems (e.g. conflicting requirements between different teams, process management difficulties, the explosion of data, limited and overworked personnel, etc.). For instance, some of the UCD actions had additional goals related to dissemination, promotion or pedagogy. Such additional layers can contribute to the complexity of UCD actions by blurring the roles within the team deploying the actions, challenging its coordination and effectiveness, and undermining the actions' end goals.

Concerning the claim that usability in API design can be excessively focused on specific design features, and fail to deliver in a more holistic way (Venners and Eckel, 2003), our experience has shown otherwise. UCD can prevent this excessive focus and support a more holistic approach, which is more suitable for the evaluation of toolkits (Wobbrock, 2012). This supports the argument of Ritter et al. (2014) regarding the broader focus and lesser reliance on formal methods of UCD when compared to other usability approaches which may be more prone to this problem.

We observed that the methodology had different degrees of acceptance with different RAPID-MIX stakeholders. Despite efforts to make sure we selected the right techniques, applied them correctly, deployed the actions effectively, and delivered useful documentation, their overall usefulness was occasionally questioned. We received contradictory remarks from stakeholders within the RAPID-MIX consortium (e.g. RAPID-MIX API designers, RAPID-MIX SME developers) about the relative contribution compared with the amount of time invested in UCD actions; or, about whether this was an adequate methodology to apply to the design of an API. In some situations, there was a lack of interest of RAPID-MIX API designers in engaging with particular techniques recommended in the UCD literature, for instance, collaboratively analysing user data, crafting user personae or doing API walkthroughs. Many of these occurrences could be identified as instances of what Holtzblatt et al. (2004) identified as the organisational backlash, or as consequences of fragmentation in the overall design process, as suggested by Norman (2013).

In RAPID-MIX, there were other and 'more-or-less' implicit approaches to the design of the RAPID-MIX API, besides UCD. This means that there was a conflation of design approaches that might have crept into the design process, and which is very difficult to determine with precision. For instance, there was also a designer-centric approach to API design (Venners and Eckel, 2003), which resulted from RAPID-MIX designers' previous experiences, i.e. both as users and as designers of APIs, including previous experience in the design of APIs in the domain of applied ML. This was observed in both RAPID-MIX API subsets based on XMM (Françoise et al., 2013) (XMM-node and mano.js subsets) and influenced by Wekinator (Fiebrink, 2011b)—which used the Weka API and inspired the RapidLib C++ and JS subsets. API design heuristics (Cwalina and Abrams, 2009; Tulach, 2008)—i.e. prescriptive guidelines or recommendations based on empirical knowledge—have also influenced the RAPID-MIX API design. Their application is usually straightforward to general API design aspects—e.g. naming consistency, readability, style conventions, conceptual domain conventions, etc.—and inevitably influence the design of any general API.

Although user-centred (Myers and Stylos, 2016), designer-centric (Venners and Eckel, 2003) and heuristics-based (Cwalina and Abrams, 2009; Tulach, 2008) approaches to API design tend to be compartimentalised and differentiated, it is reasonable to assume that, between them, there is an overlap and a varying amount of consideration for the user. Applications of designer-centric or heuristics-based approaches that prove effective for design also have to consider the user in a more or less direct way. For instance, both these approaches consider the user in an indirect way, through previously accumulated knowledge about users, or privileged designer's knowledge about the context of use. They also consider the user in a more direct way when, for instance, there is a need to validate design decisions that follow from such approaches, with users and in their context of use.

The application of UCD for lightweight formative evaluation yielded undeniably useful results. As we have shown before, both actions led to useful insights which had an impact on the development of RAPID-MIX API, with real and incremental enhancements. On one hand, a UCD approach validated some of the design assumptions, such as the abstraction level, general usability, usefulness for rapid prototyping, and how the RAPID-MIX API caters well to the opportunistic approach of creative developers who lack ML expertise. On the other hand, a UCD approach helped identify usability and technical issues, shortcomings of the documentation and learning materials, and lack of support for working more effectively with data. A UCD approach also challenged some of the design assumptions—mostly about the distinct support that should be provided for different user groups with different levels of expertise in ML and software development—in terms of the learning content, learning curve, and ceiling of the middleware.

Consider the practical effects of the objects of your conception. Then, your conception of those effects is the whole of your conception of the object..
— Charles S. Peirce

Chapter 7

Formalisation of Learning

This chapter integrates the insights and contributions of the previous action research cycles in the formalisation of learning. Section 7.1 introduces the chapter with a brief methodological review of the *Formalisation of Learning* stage of Action Design Research. Section 7.2 articulates the learning from the action research cycles with the theoretical bases that I proposed in the problem formulation stage (Section 5.2). Section 7.3 generalises from the learning obtained through the design of RAPID-MIX API to design principles for infrastructural software for user-innovation toolkits with interactive machine learning. Section 7.4 generalises from the learning through the application of Action Research and UCD in the EU H2020 -funded project RAPID-MIX to derive recommendations for consortia to lead rapid innovation actions and apply UCD.

7.1 Introduction

In Action Design Research, the stage of *Formalisation of Learning* outlines the main contributions of research through the principle of *Generalised Outcomes* (Sein et al., 2011). By adhering to this principle, the situated learning obtained through the adopted methodology is developed “into general solution concepts for a class of field problems” (Sein et al., 2011). This entails articulating the problem as an instance of a class of problems and the solution as an instance of a class of solutions. In this stage, the formalisation of learning includes the knowledge about the ensemble artefact and the utility for its users, and a set of design principles which emerged in the BIE stage (Chapter 5).

In Chapter 5.2, I formulated my research problem with a set of contributing theoretical bases. I hypothesised that Amershi’s (2011) and Dudley and Kristensson’s (2018) design spaces could be combined and articulated with von Hippel’s User Innovation theory (von Hippel, 2005) as a framework for better understanding the design of ML developer tools and the needs of developers building IML systems. The following sections articulate these theoretical constructs with the empirical findings from our studies in the action research cycles, to discuss how the UCD approach

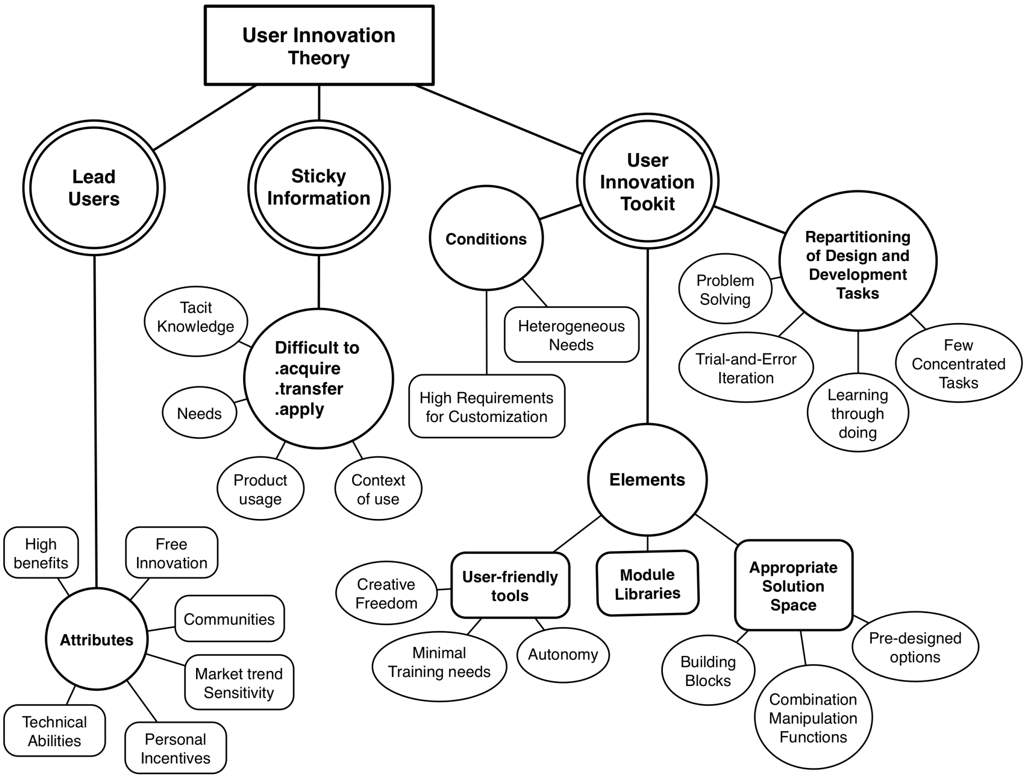


Figure 7.1: Concept map created to abstract von Hippel's (2005) theory of User Innovation

to the RAPID-MIX API can be understood as the design of infrastructural software for end-user innovation toolkits with IML.

7.2 The design of ML developer tools through User Innovation and HCML lenses

Previously, in Sections 5.2.4 and 5.4, I discussed the development of the RAPID-MIX API design space with dimensions of analysis that emerged through an informal, ad-hoc, but consensual and collaborative manner—which accords with MacLean et al. (1991). Here, I discuss how we may employ the generic UIT solution space as theoretical 'glue' to accommodate and unify Amershi's (2011) and Dudley and Kristensson's (2018) design spaces and arrive at a framework that can help us to scope and understand better the design space of ML developer tools such as the RAPID-MIX API.

The assumptions in RAPID-MIX in relation to the design of developer tools for rapid prototyping with IML bode well with the pattern of user innovation postulated by von Hippel (2005). Figure 7.1 introduces the concept map that I have created to better understand the core taxonomy of von Hippel's User Innovation theory (von Hippel, 2005) and how it is prioritised. Innovation theory comprises three main premises: (a) *Lead Users* (von Hippel, 1986), (b) *'Sticky' Information*—asymmetry of information between end-users and industrial producers that shifts the locus of

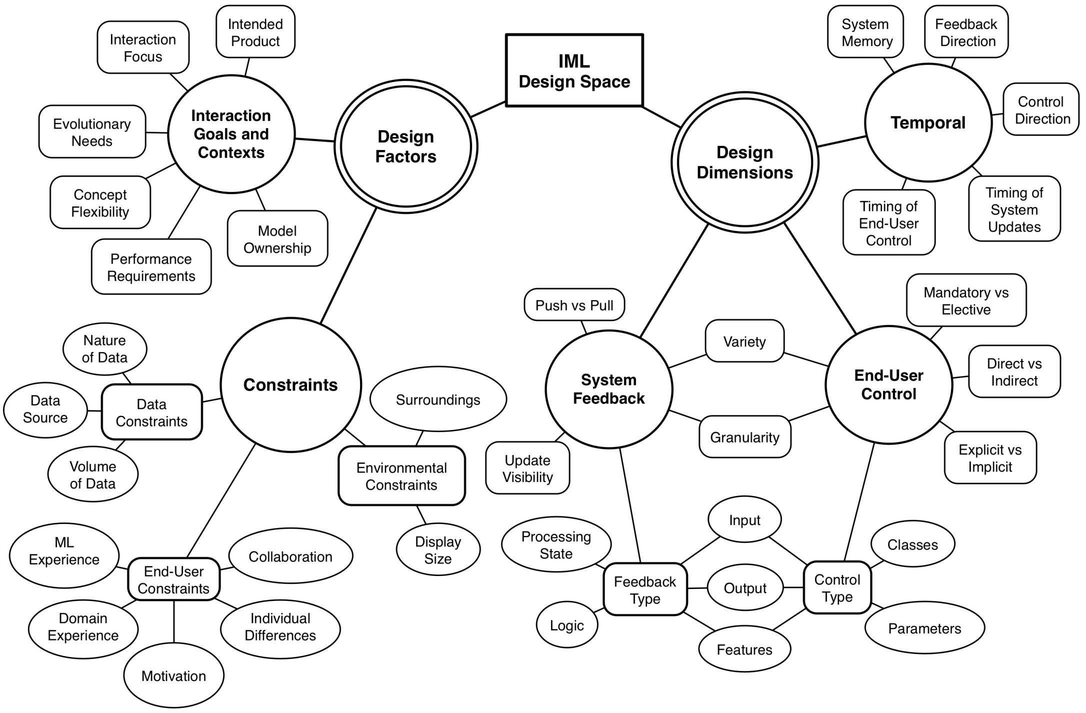


Figure 7.2: Concept map created to abstract Amershi's (2011) design space for *End-User Interaction with Machine Learning* (EUI-ML)

problem solving—(von Hippel, 1994), and (c) *User-Innovation Toolkits* (UITs) to propel user innovation (von Hippel, 2001). The elements of the concept map corresponding to these main premises and their constituent elements are congruent with theory and have been thoroughly reviewed in Section 2.2.1.

Figure 7.2 introduces the EUI-ML concept map that I have created to abstract Amershi's (2012) design space. The EUI-ML concept map shows how the taxonomy of Amershi's design space is organised and prioritised. Similarly, Figure 7.3 introduces the C-IML concept map which I have created by building upon Dudley and Kristensson's (2018) design space. The design space for C-IML breaks down IML systems structurally into a four-component architecture, and it also breaks down the interactive model-building task into four sub-tasks. The elements of both concept maps and corresponding taxonomies are congruent with theory and thoroughly reviewed in Section 2.1.2.2.

These concept maps are cognitive tools, devices for thought which show how the taxonomies are organised and prioritised. I employ these taxonomies' elements as dimensions of analysis of the design space in order to discuss some of the outcomes of the UCD actions with the RAPID-MIX API. I discuss how we may employ the generic UIT solution space as theoretical 'glue' to accommodate Amershi's (2012) and Dudley and Kristensson's (2018) design spaces and arrive at a unifying framework that can help to scope and better understand the design space of ML developer tools such as the RAPID-MIX API.

In the following sections, I discuss how these Human-Centred Machine Learning design spaces can be combined to complement each other, and articulated with User Innovation theory and the

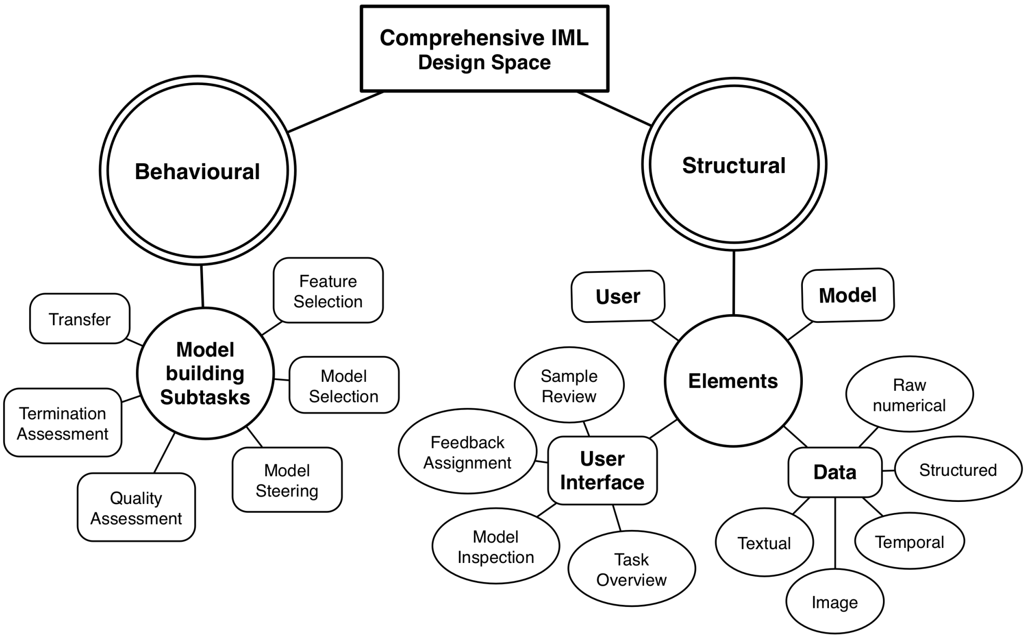


Figure 7.3: Concept map created to abstract Dudley and Kristensson's (Dudley and Kristensson, 2018) *Comprehensive IML* (C-IML) design space

insights from the action research cycles.

7.2.1 Consonance between perspectives

There are relationships of consonance, agreement or compatibility between perspectives based on von Hippel's (2005) User Innovation theory, and Amershi's (2012) and Dudley and Kristensson's (2018) design spaces. Figure 7.2.3 (pg. 228) shows the relationships that all three perspectives consider highlighted in different colours:

- User characterisation (pink highlighted)
- Goals, needs and context-dependent information (purple highlighted)
- Appropriate design space with elements of user control and feedback (blue highlighted)

7.2.1.1 User characterisation

All three perspectives consider user characterisation, as shown by Figure 7.2.3 highlighted in pink (pg. 228) and in the following concept maps' elements:

- C-IML (Figure 7.3) – user(s) as main driver(s), domain expert(s), without deep understanding of ML (capable of training, refining, or integrating a model), dynamic and unreliable, dependent on the information and guidance provided by interface.
- EUI-ML (Figure 7.2) – end-user constraints—e.g. collaboration, domain experience, individual differences, ML experience, motivation.

- UI (Figure 7.1) – lead users (innovators, ahead of market trends; able to anticipate general demand and identify specific market needs; holding technical abilities and great incentives to engage in modifying products or developing innovative solutions which can fulfil these needs).

Empirical examples

In markets with highly heterogeneous consumer needs, certain end users—"Lead Users" (von Hippel, 1986)—are highly motivated and technically capable of developing and customising products for their own needs, anticipating overall demand and specific market needs for new products. This accords with our findings in our UCD actions at Music Hack Day, eNTERFACE'17 and JUCE ML Hackathon (Sections 4.4 and 5.7) and with long-term RAPID-MIX API users, enquired with the cognitive dimensions questionnaire (Section 5.8).

Participants in our studies with the RAPID-MIX API displayed many of the attributes that von Hippel (1986) uses to characterise *Lead Users*. Among these attributes were a high degree of intrinsic motivation, customisation and development skills, the anticipation of market needs, building for their personal needs, and hobbyism. In the context of RAPID-MIX, creative coders, audio software developers, hackers, makers, and long-term users of the RAPID-MIX API, which have been abstracted to design personae in Section 5.7.5.2, correlated with von Hippel's *Lead users*.

For instance, participants at the JUCE ML hackathon (Section 5.7.1) engaged in free innovation (von Hippel, 2005) when they freely developed prototypes with the JUCE ML module (a wrapper of the RAPID-MIX API) and released them publicly on Github. These users employed the JUCE ML module—a RAPID-MIX API wrapper—to build, for instance, an embedded hardware musical application with the Bela board, and custom harmonisers with the recently released ROLI blocks. The JUCE ML module was sourced in the hackathon as part of the JUCE framework, which has many characteristics of a toolkit for user innovation, including the fostering of a user community of audio software developers.

In other UCD actions—e.g. MHD and eNTERFACE'17—we also found users of ML tools which correlated with *Lead Users*. They showed their technical abilities (including production software development in C++ and Javascript, and rapid prototyping languages, such as PureData and Max/MSP), benefits and personal incentives to innovate—e.g. learning new skills, building interesting projects and their professional reputation (Section 4.4.3). They showed market trend sensitivity by creating novel application concepts, such as musical instruments, sensor-based applications, games—e.g. building customisable gesture-sensitive musical scarfs (i.e. the MHD RAPID-MIX award winner) and web games (e.g. rock-paper-scissors at eNTERFACE).

We observed how these users held such attributes and were prone to innovate by employing ML tools for rapid prototyping such as the RAPID-MIX API. They customised solutions for themselves and for others, built end-to-end ML pipelines into their applications and trained ML models for customising application behaviour for their heterogeneous needs.

Insights

Von Hippel (2005) identified several conditions for which user-innovation toolkits (UITs) can provide the highest value to *Lead Users*. One condition is having a context in which the users of products and services may require high degrees of customisation. General-purpose machine learning algorithms enable learning of diverse concepts, making them suitable for heterogeneous user needs. IML offers a set of techniques for addressing these needs in many contexts and can support customisation and prototyping (Fiebrink, 2011b; Hartmann et al., 2007) of systems that embed ML.

In the context of RAPID-MIX, creative coders, audio software developers, hackers, makers, and long-term users of the RAPID-MIX API, who have been abstracted to design personae in Section 5.7.5.2, correlated with and displayed many of the attributes that von Hippel (1986) uses to characterise *Lead Users*. Among these attributes were a high degree of intrinsic motivation, customisation and development skills, the anticipation of market needs, building for their personal needs, and hobbyism. ML developer tools aimed at broadening and accelerating end-user innovation can build upon the motivation and the other characteristics of *Lead Users*.

Elements of User Innovation theory are consonant with EUI-ML and C-IML perspectives in relation to the application of user characterisation to the design of tools. When considering these perspectives in concert, ML developer tools should provide interfaces with information and guidance that are adequate to the ML learning experience and domain expertise of their users. They should include developers who have programming skills and other domain-specific expertise, but little to no machine learning expertise—i.e. ML -non-expert developers (Section 6.3). These concerted perspectives can raise questions about the features and tradeoffs of a ML developer tool design in relation to a specific group of target user-developers. For instance:

- Is the abstraction-level and learning style of a certain ML developer tool adequate for a certain group of developers?
- Is the documentation consistent and example use-cases comprehensive for a certain group of domain experts?
- How closely can we map the elements of ML developer tools to a specific domain of expertise?

7.2.1.2 Goals, needs and context-dependent information

All three perspectives consider goals, needs and context-dependent information, as shown by Figure 7.2.3 highlighted in purple (pg. 228), and in the following elements of the concept maps:

- C-IML (Figure 7.3) – data and features (data agnosticism is considered the ideal but difficult to achieve).

- EUI-ML (Figure 7.2) – interaction goals and contexts (concept flexibility, evolutionary needs, intended product, interaction focus, model ownership, performance requirements), data constraints (source, nature and volume of data), environment (display, surroundings).
- UI (Figure 7.1) – 'sticky information' (context of use, tacit knowledge, needs, product usage); optimal conditions for UIT (heterogeneous needs, requirements for customisation).

Empirical examples

As discussed in Section 6.2, we observed the STU contexts related to the RAPID-MIX API in practice. Users at the JUCE ML hackathon and eNTERFACE'17, were quick to learn how to employ the RAPID-MIX API in the development of software applications which learn to recognise human gestures or mimic human decision-making. Participants developed these systems for themselves, and for empowering end users who have no programming or machine learning expertise with an effective way to customise systems—i.e. systems that are trained by demonstration of gestures or decisions, such as the harmeggiator, the Leap Motion-based rock-paper-scissors game, the 3D-mesh modelling application, and the smart vintage projector controller (Sections 4.4 and 5.7).

When participants tested their RAPID-MIX API client applications and these misclassified examples, participants were able to fix them by providing corrective demonstrations to the system. They were also able to refactor the RAPID-MIX API integration code to re-design the ML pipeline and add different inputs and outputs, change parameters or the ML algorithm. For instance, when the Leap Motion-based rock-paper-scissors game misclassified the 'rock' hand stance, the developer learned to provide additional examples of that hand pose along with the correct label—'rock' label—with the intuition that the next model trained on the augmented training set would improve its performance on that hand pose.

With RAPID-MIX API users, the majority of cases for *Intended Product* were mainly in arriving at a trained ML model rather than at the results of applying the ML model. The agile approach to ML model-building which RAPID-MIX API encouraged gave the resulting ML models a transient nature. With the ease of use and speed which the IML workflow provided to client applications, RAPID-MIX API users trained models for each interaction session and discarded them after that. The primary case of *Interaction Focus* with RAPID-MIX API client end-user applications was on the workflow of the end-user applications. However, some designs limited the exposure of the IML workflow to the end-user (e.g. in rock-paper-scissors game, making the IML workflow optional and relegated to the settings panel); one design was not interactive all (e.g. FIR classifier trained in a unit test fixture).

Concerning Amershi's (2012) *Model Ownership* and Dudley and Kristensson's (2018) model building sub-task *Transfer*, the RAPID-MIX API ML model export was one feature which enabled the export of a JSON model description for transfering between applications. This feature was only used in two instances. One was a developer at eNTERFACE'17 who developed a gesture server that exported ML models to run on mobile devices. The second one was a long-term RAPID-MIX API user and SME developer which used it for a cloud-based service, which also trained models at server side and deployed to mobile phones.

Developers who used the RAPID-MIX API as proxy-users and decided to expose a full-fledged, real-time IML workflow in their end-user application instance, by exposing data collection and training functionalities and interfaces—opened up the possibility for the ML model to evolve, but also for having instability of ML model results. These decisions have implications in terms of Amershi's (2012) factors of *Evolutionary needs* and *Concept Flexibility.* These were also related to Dudley and Kristensson's (2018) model building sub-tasks, *Quality assessment* and *Termination Assessment.* Users who assessed the learning concepts according to their design goals changed their goals for ML over time—e.g. moving from 1-hand rock-paper-scissors to ambidextrous support (Section 5.7.4). Users assessing whether the training termination had been achieved, had no clear training-termination criteria. Contrastingly, two long-term users of the RAPID-MIX API enquired of with the cognitive dimensions questionnaire, considered this undesirable—e.g. concept drift, which is a problem in many different scenarios and application—particularly for production systems that require predictable behaviour.

Other factors which can relate as context-dependent information or technical needs, are Amershi's (2012) *Data Constraints* and Dudley and Kristensson's (2018) *Data* types and features. In RAPID-MIX, most of the designs fell into Dudley and Kristensson's category of *Raw Numerical* data. A few of the designs used sound and video which fall into the category of *Temporal* and *Image.* These categories also relate to Amershi's *Nature of Data* design constraints. Some RAPID-MIX API user designs employed unexpected volumes of data or number of inputs—e.g. FM synth path generator used FFT (Fast Fourier Transform) (Section 5.7.2). Other designs introduced new devices as data sources (e.g. ROLI Lightpad), or hosted ML models in new and particularly constrained systems (e.g. Bela on Beaglebone Black, which required floating-point data types to represent data). Other designs hit critical usability issues related to the inherent performance and memory limitations of the browser environment (i.e. single-thread client-side JS runtime) and the current architecture of the library when working with a high volume of data or number of models (e.g. gestural data controlling one model per vertex of 3D geometry).

Insights

From observations of the multiple STU contexts of the RAPID-MIX API (Section 6.2) rises a notion of different layers of 'sticky' information or contextual design knowledge. Such layers may be attached to different STU contexts and provide different affordances to different groups of users. RAPID-MIX API users in primary STU contexts were able to rapidly prototype software solutions to their own needs and goals and develop the knowledge required to program with an ML API. Likewise, the resulting RAPID-MIX API client applications allowed customisation to both the original developers and to their end users in secondary STU contexts. This constitutes another layer of contextual design knowledge, or ‘sticky’ information.

On one hand, ‘sticky information’ can thus refer to the design knowledge required to program a specific end-to-end ML pipeline using an ML API. On the other hand, it can refer to the knowledge required to use the ML pipeline to build custom ML models for a specific goal. ML developer tools that build on general-purpose machine learning algorithms and that encourage an IML approach, such as the RAPID-MIX API, can provide effective solutions for capturing and embedding this specialist, changing, tacit and context-dependent knowledge about an intended design. End-user applications created with ML developer tools to expose the IML workflow also enable users to capture and embody knowledge into the system.

Both the EUI-ML and the C-IML design spaces appear to correlate well with von Hippel’s ‘sticky information’, concerning the user- and need-related knowledge, including knowledge about product usage and context of use, that *Lead Users* of ML developer tools apply. We can analyse ML developer tools or API design features and their results in user evaluation against user-centred design factors of Amershi’s design space concerning user *Interaction Goals and Contexts* and *Constraints*. In particular, *Intended Product*, *Interaction Focus*, *Evolutionary Needs*, *Concept Flexibility* and *Model Ownership* and *Data Constraints*, can be applied to the analysis of ML API client applications to inform the design of ML APIs, including structure and documentation. Such factors can help to understand and extract ML infrastructural software requirements from features of ML API client applications, their construction process and usage.

When applying these perspectives to the design of ML developer tools, they lead to questions that can hint at ‘blind spots’. For instance:

- How well does an ML API support the creation of an end-to-end ML pipeline that was mapped from the interaction goals for a certain end-user application?
- What are the implications in relation to design features of ML developer tools of supporting a specific sensor as the data source of an IML application—concerning data structures, selection of algorithms, which may accommodate the nature and volume of data of such specific sensor?
- How well does an ML API support an end-user application requirement to communicate the termination criteria for a specific combination of data source and ML-algorithm?
- How should the ML API documentation convey such information and additional guidance to a certain domain-expert user?

7.2.1.3 An appropriate design space with elements of user control and feedback

All three perspectives consider a design space with elements for user control and feedback, as shown by Figure 7.2.3 (pg. 228) highlighted in blue, as well as in the following concept maps’ elements:

- C-IML (Figure 7.3) – user interface components (sample review, feedback assignment, model inspection and task overview).
- EUI-ML (Figure 7.2) – control and feedback (types, directness, comprehensibility, optionality, granularity) and timing (of control, feedback, direction, memory).
- UI (Figure 7.1) – solution space with appropriate building blocks, combination and manipulation functions, and pre-designed options; repartitioning of design and development tasks, problem solving, learning through doing and trial-and-error iterations.

Empirical examples

One particularly successful metaphor which I used in the development of RAPID-MIX API code examples (e.g. Figures 5.8 and 5.10)—and to which Jasper and Blaha (2017) refer as the DATA STREAM AS MEDIA PLAYER metaphor—used activation icons (e.g. *Play*, *Stop* and *Record*) for the control of certain aspects of the IML workflow, particularly the model-building sub-tasks *Model Steering* and *Termination Assessment* (Dudley and Kristensson, 2018). Participants at the 24-hour ANVIL Hack III found this design very engaging. At eNTERFACE'17, participants adopted this metaphor to understand the IML workflow, which appeared to engage and motivate the adoption of the RAPID-MIX API through those code examples.

Participants used the GUI controls with this metaphor as proxy-objects for understanding RAPID-MIX API classes, and for using them as building blocks for their own projects. They extended or adapted these objects for own their needs. However, such a well-defined metaphor does not preclude developers of end-user applications and proxy-users of RAPID-MIX API from determining how the IML workflow should be in their end-user application. The ML developer tools proxy-users might decide to design a domain-specific metaphor for the end-user application interface or expose the IML workflow in an idiosyncratic way, as in the cases of the FIR classifier and rock-paper-scissors game.

In such cases, developers chose to selectively obscure certain controls and parts of the IML workflow, for instance, the data collection, parameter setting or the algorithms selection controls. This happened, for instance, with the participant building the rock-scissors-paper application at eNTERFACE'17, who integrated the IML workflow as part of a second-level configuration settings panel of his application.

Insights

As discussed in Section 6.2, RAPID-MIX API client applications with the IML workflow can support a shift of design activity to end users in a secondary STU context, which accords with the postulates of User Innovation theory. Von Hippel also notes the importance of UITs in facilitating the translation of users' designs into production, which he suggests that it may be beneficial to provide IML users with easy ways to use trained models outside of prototyping environments. RAPID-MIX API features cross-platform characteristics and trained ML models exports, that relate to this translation by facilitates embedding ML functionality into users' software or hardware projects.

ML developer tools that provide adequate consideration for ML -non-expert developers have a high abstraction level, support opportunistic approaches, and are closely mapped to their domain of expertise. They support an autonomous and incremental learning style, with a smooth learning curve, and help the ML -non-expert developer to progress in their learning experience. They facilitate the learning and an effective developer experience through consistent documentation, contrasting examples with practical use cases of interest to prospecting users. ML developer tools should encourage rapid prototyping of end-to-end ML pipelines and IML workflows as a first approach, where developers train and refine, or integrate a model. They support different kinds of user, including more skilled and experienced developers, by providing more flexibility, and a penetrability which is sensitive to context and to the users' implementation needs.

7.2.2 Contrast between perspectives

When considering a unified framework it is useful to understand what unique contributions each of these three perspectives makes and what are the contrasts between them. Such analysis can reveal useful to the distillation of concrete design guidelines and principles of infrastructural software and developer tools for user-innovation toolkits with IML. However, contrasts between these three perspectives is mostly related to absence, level of specialism, completeness, or granularity, rather than through opposition or dissonance.

7.2.2.1 Contrast of C-IML in relation to EUI-ML to UIT

There are two main elements of contrast of C-IML in relation to EUI-ML and UIT. The first is the strong focus on the design of the user interface. The second one is the high level of domain abstraction and specialisation of the UI controls and model-building sub-tasks for IML, which does not exist in the other two perspectives as they are more generic. The C-IML perspective specialises the concept of building blocks, combination and manipulation functions, and pre-design options of User Innovation theory. A reframing of the C-IML perspective can suggest how ML developer tools should be designed and modularised to fulfil the needs of developers building IML applications, particularly for high-level UI components and workflow, as well as applicational building blocks.

As Dudley and Kristensson (2018) concede, an IML system might be designed for end users to focus on one or more specific model-building tasks, and to restrict end-user interaction to these tasks. Developers applying ML developer tools to the design of an IML system should consider how to design interaction for the model-building sub-tasks, and make tasks and task constraints clear and understandable so that the system enables the achievement end-users' goals. Illustrative examples of questions which follow from the application of Dudley and Kristensson's (2018) C-IML design space to understand the requirements of potential RAPID-MIX API client application features could be:

- What are the different implications of the features of a ML developer tool if employed in the development of, for instance, a *model-steering* user interface?
- Does the ML developer tool provide support for implementing a *sample review* interface in a custom application to control, for instance, the lighting rig of a stage?
- How easy does a ML API make the developer's task of training an ML model for a specific concept?

When applying the EUI-ML and C-IML in concert more focused questions can be asked, such as:

- What features of ML developer tools can provide for end-user application requirements for the *model steering* sub-task, in terms of timing of system updates or granularity of ML model feedback?

7.2.2.2 Contrast of EUI-ML in relation to UIT and C-IML

As previously mentioned, Amershi's user-centred design factors and dimensions provide a very granular and comprehensive set of elements concerning contextual information, design constraints and design dimensions for ML in general. Although they relate to User Innovation and C-IML design space, it is the granularity and comprehensiveness of the EUI-ML elements which provides the contrast to both the abstract 'Sticky Information' of User Innovation theory, and the more high-level and superficial structural characterisation of C-IML's *User* and *Data*.

Amershi's EUI-ML design space can complement the C-IML design space at a more granular level, in terms of understanding how ML developer tools may provide useful features for the design of these composite UI elements for ML model-building, concerning the affordances for system feedback, end-user control and temporal aspects. Furthermore, the *Interaction Goals and Contexts* and *Constraints*—in particular, *Intended Product*, *Interaction Focus*, *Evolutionary Needs*, *Concept Flexibility* and *Model Ownership* and *Data Constraints*, can be applied through reframing to the analysis of ML API client-application features to inform the design of ML APIs developer tools, including structure and documentation.

The observations about the C-IML and UIT concept maps suggests that Dudley and Kristensson's (2018) design space elements can complement Amershi's at a high level. The C-IML can help the identification of composites of UI elements that can extend Amershi's design dimensions.

The application of this perspectives in concert to the analysis of requirements for ML developer tools can lead to questions and help to determine design directions for ML developer tools. For instance:

- What are the implications of a developer goal, such as integrating a specific sensor as a data source into an ML application, in the design of the capabilities of an ML developer tool?

- What should a toolkit for rapid prototyping with IML, such as the RAPID-MIX API, make available to support high-data throughput or high-resolution sensors?

- Moreover, how should these toolkit capabilities be presented to the end user, considering a potential multitude of interaction goals and user-developer constraints?

- How do the needs and goals of end users of ML developer tools in terms of persistence and re-usability of ML models, determine features of those ML developer tools?

7.2.2.3 Contrast of UIT in relation to C-IML and EUI-ML

Von Hippel (2001) discusses the importance of UITs in shifting innovation to users by repartitioning design and development tasks and providing a few concentrated design tasks. In User Innovation theory, these tasks are abstract and generic. The C-IML design space proposes a set of well-defined and specialised set of model-building sub-tasks that an IML system may encompass—i.e. feature selection, model selection, model steering, quality assessment, termination assessment and transfer (Dudley and Kristensson, 2018). However, this set of well-defined, high-level or otherwise specialised tasks is perhaps implicit, if not absent in Amershi's design space.

The affordances of general-purpose ML algorithms, and particularly, of the IML workflow, can be understood as supporting the repartitioning of design and development tasks between multiple STU contexts. As discussed in Section 6.2, ML developer tools client applications with the IML workflow can support a shift of design activity to end users in a secondary STU context, where users train and refine an ML model. Such tasks can potentially complement Amershi's design space with a new dimension or new design factors. Particularly, as it may be observed in the EUI-ML concept map (Figure 7.2), the model-building sub-tasks may extend specifically Amershi's taxonomy elements of *Interaction Goals* and *Interaction Focus.*

Von Hippel also notes the importance of UITs in facilitating the translation of users' designs into production. This suggests that it may be beneficial to provide IML users with easy ways to use trained models outside of prototyping environments. This also correlates with Dudley and Kristensson's (2018) model-building sub-task *Transfer.* This might be implicit in Amershi's (2012) *Model Ownership* but it is not considered a design factor *per se* in the EUI-ML design space.

RAPID-MIX API features cross-platform and multi-target characteristics, and trained ML models exports. These are features of portability, which can relate to von Hippels' principle of translating user designs into productions. Model exports facilitate embedding ML functionality into users' software or hardware projects through ML tranfer learning. The cross-platform and multi-target allows the use the RAPID-MIX API to program end-to-end ML pipelines in the users' environments and deployment in their desired production environments.

Other aspect that User Innovation theory contrasts with EUI-ML and C-IML is the community aspects of the tools in general, including shared knowledge and open source licensing, which have a great impact in their adoption. These aspects will be explored in more detail in Section 7.3.2.

7.2.3 Insights and implications to theory

The design of ML developer tools can be led in such a fashion that it may provide a meaningful coverage of User Innovation theory, constructs and principles. User Innovation principles may translate well to design features of ML infrastructural software and developer tools for rapid prototyping with IML. The design of RAPID-MIX API, in particular, correlated well with the requirements and elements of a UIT (von Hippel, 2001). The design of RAPID-MIX API aimed to maximise autonomy, minimise the need for specialised training, and reinforce the creative freedom, by allowing users to focus on their design goals and on the ML functionality they want or need to integrate into their projects. "Sticky information" (von Hippel, 1994) ingrained in an end-user application design with the RAPID-MIX API, can be understood as the developer's craft of a ML pipeline, with a configuration of inputs and outputs of this pipeline that is integrated within her application to support a real-time IML workflow.

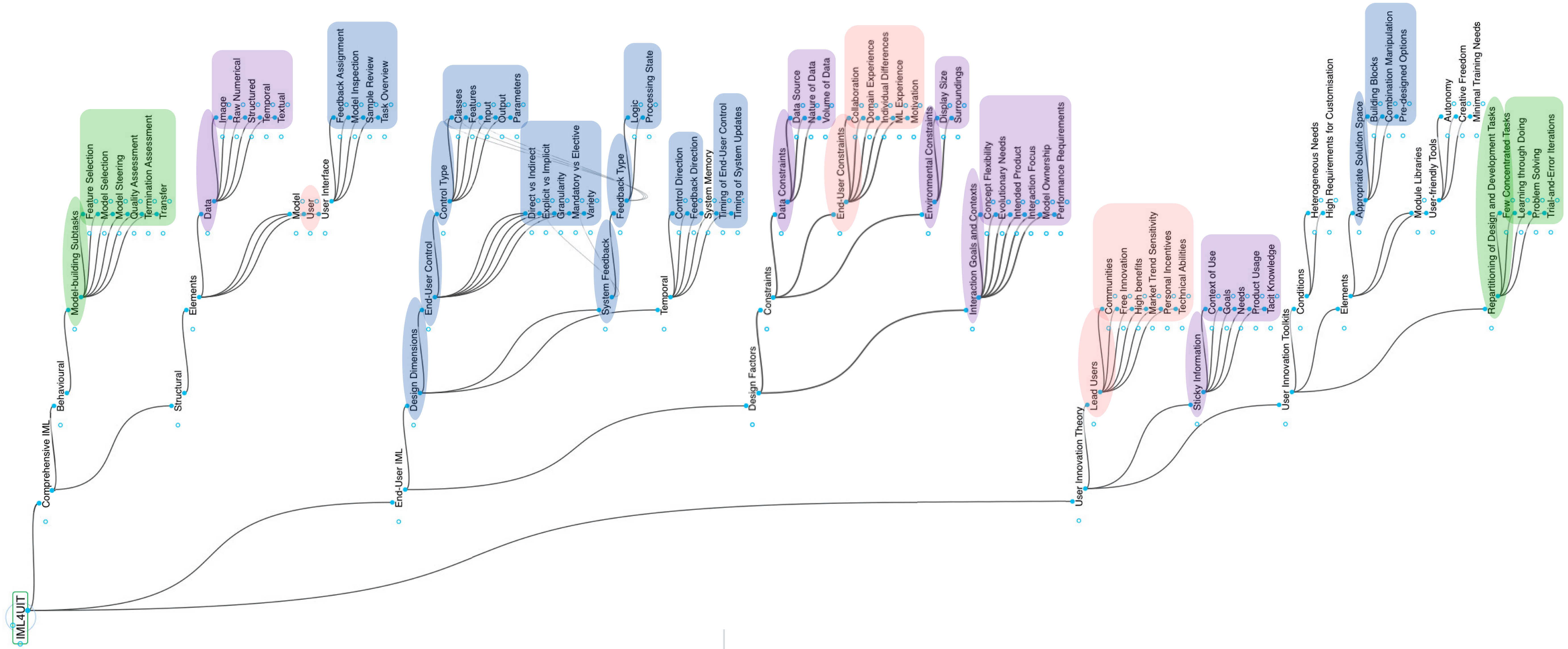
As noted in our evaluation studies with users of the RAPID-MIX API, which focused on the usability and user-friendliness of ML developer tools, they can provide creative freedom and a good autonomy to groups of ML non-expert developers which correlate with Lead Users. ML developer tools such as the RAPID-MIX API, which show many of von Hippel's (2001) attributes for UITs, including empowering users to employ their skills, focus on the design tasks, and defer extensive specialised training, can be understood as artefacts designed to "democratise" machine learning. They can broaden and accelerate end-user innovation and make the benefits of learning algorithms realisable by a wider range of people, including people without machine learning expertise.

Analogously to the LSI software toolkit, the seminal UIT that disrupted the integrated circuit industry (von Hippel, 2001), in ML developer tools there can be a repartition of design and development tasks. Development tasks, in the sense of von Hippel (2001) refer to the efforts of ML developer tool designers to embed ML design knowledge through the curation of ML algorithms, distillation of workflows, architectural design, API and developer-facing interfaces design and usability. Design tasks, refer to what ML developer tools expose and provide to developers as lead users that are eager to create and innovate with ML. With IML design for toolkits, developers leverage a workflow, libraries of modules (von Hippel, 2001) for ML and DSP algorithms, express their needs through programming, problem-solving, and learning through doing and trial-and-error.

Most interestingly, in our user studies with ML infrastructural software, we found a pattern that suggests a potential expansion to User Innovation theory, or at least, further research. On one hand, users developed thin wrappers of the RAPID-MIX API as modules for environments such as JUCE, Max/MSP, Unity, React, etc. On the other, they developed "second-order" middleware with embodied domain-specific semantics and design knowledge to better adapt them to their design needs and of others. As discussed in Section 6.2, this pattern consists of instances of re-tooling where *Lead Users* appropriated of the RAPID-MIX API to adapt it to other designs, development environments and user-innovation toolkits, or to create second-order toolkits. Alternatively, this also suggests new uses for IML in creating or improving UITs across a variety of application domains.

I argue that we can employ von Hippel's (2005) generic UIT solution space as theoretical 'glue' to accommodate and unify the HCML design spaces and arrive at an adequate working theory to explore, inform and analyse the design of interactive machine learning for user innovation toolkits. User Innovation theory in concert with the HCML design spaces can provide an adequate framework to explore, analyse and informe the design of developer tools for rapid prototyping with interactive machine learning which aim to support end-user innovation. This working theory can help us understand better the design space of ML developer tools such as the RAPID-MIX API.

Moreover, this working theory can be beneficial to the analysis and reflection about a user-centred approach to the design and evaluation of developer tools for rapid prototyping with IML. Such a theory may provide an effective way to guide method to design and run studies, and analyse their outcomes to obtain insight and understand the nature of the problem of designing interactive machine learning for toolkits for end-user innovation. A framework which unifies User Innovation theory with the HCML design spaces can be employed, as a working theory that helps to understand and work more effectively in the context of projects related to technology transfer and end-user innovation, such as RAPID-MIX.


IML4UIT
Comprehensive IML
Behavioural
Model-building Subtasks
Feature Selection
Model Selection
Model Steering
Quality Assessment
Termination Assessment
Transfer
Structural
Elements
Data
Image
Raw Numerical
Structured
Temporal
Textual
Model
User
User Interface
Feedback Assignment
Model Inspection
Sample Review
Task Overview
End-User IML
Design Dimensions
End-User Control
Control Type
Classes
Features
Input
Output
Parameters
Direct vs Indirect
Explicit vs Implicit
Granularity
Mandatory vs Elective
Variety
System Feedback
Feedback Type
Logic
Processing State
Temporal
Control Direction
Feedback Direction
System Memory
Timing of End-User Control
Timing of System Updates
Design Factors
Constraints
Data Constraints
Data Source
Nature of Data
Volume of Data
End-User Constraints
Collaboration
Domain Experience
Individual Differences
ML Experience
Motivation
Environmental Constraints
Display Size
Surroundings
Interaction Goals and Contexts
Concept Flexibility
Evolutionary Needs
Intended Product
Interaction Focus
Model Ownership
Performance Requirements
User Innovation Theory
Lead Users
Communities
Free Innovation
High benefits
Market Trend Sensitivity
Personal Incentives
Technical Abilities
Sticky Information
Context of Use
Goals
Needs
Product Usage
Tacit Knowledge
User Innovation Toolkits
Conditions
Heterogeneous Needs
High Requirements for Customisation
Elements
Appropriate Solution Space
Building Blocks
Combination Manipulation
Pre-designed Options
Module Libraries
User-friendly Tools
Autonomy
Creative Freedom
Minimal Training Needs
Repartitioning of Design and Development Tasks
Few Concentrated Tasks
Learning through Doing
Problem Solving
Trial-and-Error Iterations

7.3 Design principles of Interactive Machine Learning for User-Innovation Toolkits

In the following sections, I discuss the design of the RAPID-MIX API concerning the underlying assumptions, decisions, evaluation with UCD actions, and emerging principles.

7.3.1 Real-time interactivity between user, data and ML algorithms

A central concern to RAPID-MIX's goals was the creation of new machine learning tools to support developers in the creative industries. The decision to focus the RAPID-MIX API design on supporting an IML approach entailed gathering consensus and reconciling the differences and overlap between IRCAM and Goldsmiths' independent but complementary approaches (Caramiaux et al., 2014; Fiebrink et al., 2011; Françoise et al., 2014). A consequence of the RAPID-MIX design process was the identification of points of uniqueness and core features of a joint approach for the RAPID-MIX API.

This joint approach—involving "fluid, easy, interaction between user and algorithm" and "end-user curation and modification of training examples" as voiced by Fiebrink (Section 5.5.2)—focused on providing real-time interactivity between the user, data, ML algorithms and models. The RAPID-MIX API intended to support developers as "users" of machine learning tools—including developers having different skill levels in application and interface development, and a non-expert level of proficiency in ML—as "end-user developers", developing systems for themselves, and as "proxy-users" developing real-time IML systems for end users.

7.3.1.1 Curation of ML algorithms

Most notably, the design decisions and lessons learned from Rebecca Fiebrink's work with Wekinator (Fiebrink et al., 2011; Fiebrink, 2011b) contributed to informing the unique real-time IML workflow that RAPID-MIX API intended to support. One instrumental contribution from Fiebrink's work to support the real-time IML workflow in RAPID-MIX API was the curation of ML algorithms based on specific affordances—e.g. low training time, low number of training examples required, high learning capacity, ability to steer model behaviour with training data sets and parameters (Fiebrink, 2011b).

For instance, the kNN algorithm may be a more desirable for supporting real-time IML approach than a Support Vector Machine algorithm, which works better with larger data sets. However, the kNN algorithm has an aggravated performance penalty when used with datasets with a large number of features (i.e. curse of dimensionality). IRCAM's XMM family of algorithms (Françoise et al., 2014) and GVF (Caramiaux et al., 2014) are compatible with these criteria and share many of the affordances as mentioned earlier.

The curation process of ML algorithms that was intrinsic to the RAPID-MIX common approach to ML API design contributes with a congruent selection of ML algoriths to effectively support a real-time IML workflow. Different ML algorithms have different design trade-offs (concerning usability, training cost and performance) that need consideration. The ML algorithm curation process can be understood as designer knowledge that RAPID-MIX API designers embodied in the tool.

7.3.1.2 A faster, intuitive and experiential approach to ML

Another important design decision for supporting the real-time IML approach was favouring direct observation of an ML model's behaviour on new data, over traditional ML model evaluation techniques, such as training accuracy and cross-validation. Supporting this design decision in the RAPID-MIX API were the lessons learned with users of Wekinator and claims from Fiebrink et al. (2011). Direct observation provides an effective way to assess a model's performance and can include subjective metrics, such as correctness, cost, the shape of the decision boundaries, confidence, which may be more meaningful and valuable to the end user (Fiebrink et al., 2011).

A real-time IML workflow, as encouraged for end-user applications leveraging on the RAPID-MIX API where the end-user creates, curates, and modifies training data iteratively to build ML models, and steer their behaviour based on direct observation, may yield a smaller gulf of execution and evaluation (Norman and Draper, 1986). Such an approach can support faster development of intuition about how to work with ML algorithms and understand their behaviour, which may be more attractive to novices. It can also support opportunistic approaches to system development that are more aligned with the goals of rapid prototyping.

The decision of carrying the benefits of a real-time IML approach onto the RAPID-MIX API design can be justified by the assumptions about the useful features it provides to digital design and software development contexts. The speed, directness, and iterative nature of interaction involved in IML are known for facilitating rapid prototyping and iterative refinement activities that are important to the design of new systems. IML can reduce the time needed to instantiate and modify prototypes, and consequently, facilitate the exploration of the design space. For instance, IML has been found useful in creative applications in which ML is used to design interactions with sensors, audio, or video data—e.g. Wekinator (Fiebrink et al., 2011), Exemplar (Hartmann et al., 2007).

By supporting an IML approach, the RAPID-MIX API would allow developers to build new systems by demonstration, resulting in a faster and more intuitive development process. When the goal is to develop a system that responds to complex phenomena, the developer cannot easily describe in code heuristics (e.g. high-dimensional or noisy sensor data, or embodied interactions), IML can also result in systems that are more accurate in their labelling or response to new data. Such an experiential approach maximises the interaction between the ML model and the end-users experience of the ML model's behaviour, and can facilitate ML teaching and learning (Sarkar, 2016).

IML can be understood as a tool for end-user programming and systems customisation that provides a unique, user-centred, direct approach. It can also be understood as a way to "democratise" machine learning, making the benefits of learning algorithms realisable by a wider range of people. As infrastructural software and a developer toolkit, the RAPID-MIX API intended to carry the benefits and affordances that an IML approach and workflow can provide, to the hands of developers as end-users and as proxy-users to support the development of IML-based end-user applications, and ultimately, end users.

7.3.2 Scaling IML systems development with user-centred infrastructural software

Infrastructural software enables the scaling of common benefits across many client applications and therefore, delivering more powerful results. As a primary concern and core design goal of the RAPID-MIX consortium stakeholders—as early UCD actions have shown (Section 4.3.5.4)—the RAPID-MIX API intended to maximise the power and value of a common infrastructure to leverage real-time, multimodal IML workflows across different applications, development and deployment environments, and different user skills. Contributing to this core design idea were underlying design principles such as:

- Modularity – the value of modules to end users of a toolkit, according to von Hippel (2001), is their rapid integration in custom designs, which allows end users to focus on the novelty of the design for their specific problem. In software design, modularity is a general and central aspect, related to encapsulation and abstraction from implementation details (Fowler, 2004). The RAPID-MIX API provided software components as independent, self-contained, and interoperable modules which could be mixed and matched in different configurations of ML pipelines, and therefore, support many different applications. It includes components for the different stages of the generic ML pipeline, including DSP routines and ML algorithms (see Section 5.6.2). Interoperability is intrinsic to modularity in the RAPID-MIX API as it provided for interoperable modules in two different ways. First, the RAPID-MIX API implemented interoperability through an architectural design which involved independent modules in a bundle and employed the Adapter pattern to provide a common high-level abstraction layer. This architecture is dependent on versioning systems and code repositories mechanisms, such as *Git Submodule*, which imports the independent modules to a standard solution for building a binary able to be deployable in users' projects. Second, the RAPID-MIX API supported component interoperability through the integration of protocols, the definition of formats and the implementation of supporting services—i.e. using Websockets and OSC protocols, and defining a specific JSON format for the Repovizz RESTful API.

- Portability – ML infrastructural software that supports application development in different environments (cross-platform) and deployment in different devices and environments (multi-target) has certain advantages, such as greater reach and penetration, the reduction of development and maintenance costs, and uniformisation of design features. We experienced these advantages mainly when RAPID-MIX API users had heterogeneous needs. Notably, these advantages were significant, not only when the RAPID-MIX API was used as user-innovation toolkit, but also as a way to integrate with and innovate other user-innovation toolkits. This was requested in the preliminary studies by potential end users and RAPID-MIX stakeholders. The RAPID-MIX API as toolkit of infrastructural software provides a way to integrate ML pipelines and IML workflows in end-user applications. As cross-platform and multi-target ML infrastructural software, the RAPID-MIX API was used to power other software toolkits, with different levels of abstraction and different programming paradigms. The dual-layer architecture of the RAPID-MIX API reflects its portability. There were different factors motivating each layer, such as:

  - C++ API layer – the provision of the RAPID-MIX API in a C++ code base was congruent and coherent with different factors such as:

    a) prior software design decisions in background RAPID-MIX technologies—e.g. Maximilian (Grierson, 2010), XMM (Françoise et al., 2014), JUCE—concerning adoption of C++ for their code base.

    b) the intrinsic advantages of the C++ language—e.g. general purpose, intermediate abstraction level, performance, and most significantly, portability, which was of great importance for the RAPID-MIX stakeholders' vision, use cases, products and user needs.

    c) the familiarity with, and preference for the C++ language of users from target groups—as reported by the findings of UCD actions, developers, in particular, of audio software, embedded hardware and mobile applications, and creative developers working with development frameworks, such as Openframeworks and Cinder.

  - JavaScript API layer – the provision of the RAPID-MIX API with a Javascript layer was congruent and coherent with different factors such as:

    a) prior software design decisions in background RAPID-MIX technologies —e.g. Maxim.js, XMM-node, XMM-lfo and CodeCircle (Fiala et al., 2016).

    b) the intrinsic advantages of Javascript, such as being easy to distribute and deploy; portability—both in server-side, running in node.js, and client-side running in desktop and mobile browsers; widely adopted as a programming language that is relatively easy for novice programmers;

    c) the extremely low concentration of Javascript ML APIs, which provided an attractive innovation opportunity by contributing to the "push" of ML into the browser. A survey that I did at the time about existing ML solutions—between libraries, frameworks, cloud services and B2B products—showed that the majority of ML implementations used Python, C++, Java and .NET. Two of the very few precursors implementing ML in Javascript were Synaptic[1] and Brain.js[2].

d) the opportunity to acquire the "low-hanging fruit" of ML for Javascript via transpilation techniques, which provided an easy and effective solution for engineering cross-platform libraries by porting C++ to asm.js via using emscripten (Zakai, 2011; Zbyszyński et al., 2017).

- Open source – as an OSS project with an MIT license[3], RAPID-MIX API code was made freely available. Adopting OSS as development and licensing strategy also contributed to the scaling of reach of the RAPID-MIX API to developer-users. As with any other OSS, the costs of distribution of RAPID-MIX are mostly negligible, which also reduces the cost of programming a new solution (Fowler, 2004; Lakhani and Von Hippel, 2003). This contributed to empower users to reuse the ML algorithm implementations provided by RAPID-MIX API. The direct costs of OSS ML infrastructural software as the RAPID-MIX API are mainly the amount of time needed to learn how to use a simple and minimal API.

- User-centredness – this attribute encompasses user-friendliness (i.e. easy to learn and easy to use) and user diversity (providing for the needs of different types of users), which I identified as requirements in previous UCD actions, from both stakeholders' vision and potential users needs and goals. As ML infrastructural software for IML developed using a user-centric approach, the RAPID-MIX API provided users with different profiles the ability to build end-to-end ML pipelines and leverage on IML. It supported the needs and goals of developers with different levels of programming skills and lack of ML proficiency. The scaling power that RAPID-MIX API provides is also linked to its effectiveness at reducing the viscosity and skills barriers for IML systems development. In the next section, I discuss how this aspect provides for quicker and broader adoption.

Concerning the last point, the RAPID-MIX API design and development process involved some of the challenges that Edwards et al. (2003) identified for UCD for infrastructural software. For instance, the level of indirection associated with infrastructural software, made UCD of the RAPID-MIX API challenging, particularly in defining design and evaluation criteria and providing a feature set that strikes the right balance between usefulness and complexity. From a UCD perspective, determining which infrastructural features should go into an end-user application and how well these features address the needs of users, determines the usefulness of the infrastructural software and ultimately its worth (Edwards et al., 2003) and the same applies to infrastructural software for supporting IML.

[1] Synaptic, an architecture-free neural network library for node.js and the browser, https://github.com/cazala/synaptic

[2] BrainJS, Javascript Neural Networks library, https://github.com/BrainJS/brain.js

[3] The MIT License, https://opensource.org/licenses/MIT

Design feature definition for the RAPID-MIX API happened both indirectly and directly; it happened indirectly with the reuse of Wekinator's (Fiebrink, 2011b) design assumptions and previously learned lessons from its users. It also happened directly with the user-centric approach to the RAPID-MIX API design, which focused on implications for usability and usefulness of the design decisions of the developer-facing interface and implementation of the RAPID-MIX API, using the findings from UCD actions.

The approach taken for the RAPID-MIX API design converged with other recommendations from Edwards et al. (2003). This included, for instance, the use of multiple scenarios to understand the common infrastructural needs of RAPID-MIX consortium stakeholders and end-users and drive design (Section 4.3.5.4). Or, for instance, starting the design process of the RAPID-MIX API providing a minimalist set of features for supporting the interactive machine workflow, gradually incrementing the complexity, with care to avoid bloat. We also produced a set of demonstrators as core test client applications of the RAPID-MIX API in order to support the process of validation and assessment of infrastructure code features. These demonstrators represented the class of applications that the RAPID-MIX API was intended to support, and provided users examples and building blocks for integration of a real-time IML workflow in their applications, which would be used by their end-users.

Edwards et al. (2003) claimed that the value of infrastructural features provide is tied to how they are expressed through client code applications, and how the client application become extended with new technical capabilities or more usable interaction styles. The RAPID-MIX API demonstrators followed UCD guidelines and their design and development was also guided for usability and usefulness to demonstrate how to build ML pipelines and integrate the IML workflow. This was a resource-intensive process which was distributed across RAPID-MIX partners. It included independent cycles of requirements gathering, prototyping, evaluation, and iteration as RAPID-MIX partners developed prototypes with the integration of the RAPID-MIX API with MIX products concurrently.

Appropriating Buxton's famous title, UCD of infrastructural software is about "getting the right" infrastructural features and designing them in the "right" way, just as much as with other UCD artefacts, such as full-fledged GUI applications.

7.3.3 Reducing the viscosity and skills barriers to ML developer tools

In Section 7.3.1, I discussed the benefits that an IML approach can provide, mainly, of the real-time, experiential, direct, and user-friendly nature. As Olsen (2007) notes, a toolkit can provide value in different ways. It is essential, therefore, to distinguish between the design assumptions and features that relate to the real-time IML workflow and the design assumptions and features that relate to the RAPID-MIX API, as a developer tool designed to support and encourage such a specific approach to ML.

Dudley and Kristensson (2018) argued that interface design is critical to the success of the IML process. This argument was supported by the promising results of the IML workflow with high-level GUI interfaces, which emerged in the context of HCML research. Essentially, these interfaces combine the adaptive nature of ML algorithms—typically hand-coded from “scratch”, or provided by ML infrastructural software and toolkits such as the RAPID-MIX API—with the craft of interaction design, to support a co-adaptive process, in which the end user steers the ML model behaviour. As Amershi (2012) argues, the effectiveness of the interface design that supports the interaction between the end user and machine learning, and the extent to which it exposes the right control and feedback to the user will determine the success and the importance of the end-user IML application.

Given the close relationship between developer-facing infrastructural interfaces (or APIs) for ML and end-user ML interfaces, it is straightforward to assume that the former, an ML developer tool, will have an impact on the latter, the high-level end-user ML interface and outcome of the application of the developer tool. However, both Dudley and Kristensson (2018) and Amershi (2012) argued that there is a lack of established principles and proven techniques for designing interaction with machine learning. The goals of understanding and improving user experiences with machine learning should, therefore, apply when the interface employed by the user is a developer-facing one, or APIs of ML infrastructural software, such as the RAPID-MIX API.

In RAPID-MIX, there was an intentional pursuit for the cumulative effect resulting from the combination of two user-friendly and pragmatic approaches—the RAPID-MIX API as a pragmatic approach to IML systems development, and the IML workflow as a pragmatic approach to the application ML learning techniques. As a developer-facing interface (i.e. API), the RAPID-MIX API provided a user-friendly and programmatic approach that facilitated rapid prototyping with infrastructural software and reduced the time to create functional end-to-end ML pipelines in end-user prototypes. The RAPID-MIX API not only supported and encouraged the IML approach, but also aligned with its design assumptions.

Another central and convergent RAPID-MIX API design goal was to provide a pragmatic and user-friendly development tool for learning ML—i.e. easy to use, easy to learn. The RAPID-MIX API facilitated a practical approach to the development of skills and intuition to ML, with self-learning and immediate engagement with real tasks. These tasks focused on programming simple real-time ML pipelines and executing the IML workflow for model-building—creating ML models and steering its behaviour, by iteratively creating, curating, and modifying training datasets.

As this thesis claims, a pursuit for principles for designing interaction with ML should consider therefore the affordances of ML infrastructural software. These affordances underlie the end-user ML interface—concerning how they constrain or empower the developer’s assumptions and goals. Rather importantly, these affordances encourage or determine a style of interaction which the developer, as a user of the RAPID-MIX API and proxy-user of the IML workflow will produce for secondary STU contexts, both for end-user applications and second-order infrastructural software.

I was able to extract attributes which contribute to the principle of reducing viscosity and lowering the skills barriers with ML developer tools, from insights provided by the RAPID-MIX API design guidelines, development process, and empirical users evaluation. They include for instance:

- Simplicity and minimalism – a programmatic approach that reduces both the complexity and the number of possibilities of setting up ML pipelines and working with ML algorithms, maximises the immediate engagement with the ML algorithm, both programmatically and via the IML workflow. Such attributes in ML developer tools can be observed, for instance, in the minimal surface area of the RAPID-MIX API, with a minimal ML class hierarchy (e.g. classification, regression), the minimal amount of class methods (e.g. record(), train() and run()), and minimal number of parameters with useful defaults. Additionally, the documentation of the RAPID-MIX API associated these design features to the most meaningful target design tasks semantically (for instance, by showing how to apply the regression object with default parameters to designing and controlling the parameters of a customisable synthesizer). Creative ML-non-expert developers were able to quickly adopt and successfully use the RAPID-MIX API as they only had to become familiar with an information structure with these characteristics.

- Example-based learning – code examples provide scaffolding (Nasehi et al., 2012) for ML-non-expert users to learn about building ML pipelines and using the IML workflow autonomously. RAPID-MIX API provided an extensive set of demonstrators and code examples associated with the tasks and use cases that were developed from early scenarios and our studies gathering user feedback and expectations (Section 4.6). Example-based approaches have been used previously in HCI research to support learning and motivate the adoption of ML tools–e.g. for real-time ML, where experts develop examples for novices to appropriate, to make use of and extend (Mellis et al., 2017), and with high-level interfaces for musicians (Fiebrink et al., 2011). It is also a standard approach in creative coding communities, such as OF, P5, Cinder, etc. Code examples were used as building blocks by users at eNTERFACE'17 (Section 5.7.4) and long-term RAPID-MIX API users.

- Standardisation – this principle had a two-fold implication in the RAPID-MIX API. On the one hand, it refers to limiting the use of specific ML terminology—e.g. Kohavi and Provost's (1998) *Glossary of Terms*—in RAPID-MIX API classes and methods to the minimum, while still conforming with standard terms where inevitable. For example, understanding the meaning of classification as a design task and machine learning task may lead to the further pursuit of what Domingos (2012) called "folk wisdom" of ML. From a UCD standpoint, this principle inherits from a general principle—i.e. standardisation for improved memorability (Norman, 2013)—which claims increased usability by having to learn about a term only once and thus lowering the barrier to participation. On the other hand, it refers to the standardisation of the different ML algorithms to a common interface and simplified conceptual model with design tasks and data types (Figure 5.6). As discussed earlier, the RAPID-MIX API implemented this standardisation through the adapter pattern (Gamma et al., 1994).

- Online development environments with documentation and interactive tutorials – users of the RAPID-MIX API had only to access functional and example-based online documentation, which mapped conceptual ML learning to interesting use cases in the creative industries domain using interactive tutorials and examples. Moreover, ML developer tools were easily accessible on an online interactive programming environment, such as CodeCircle, which provided documentation and commented examples. This provided users with immediate engagement with ML algorithms, ML pipelines and IML workflow, and immersion in the tasks they support.

As observed in our user studies, users provided very positive feedback overall about the process speed and relative quality of results obtained with the RAPID-MIX API. The accumulated effect of developing an ML pipeline with the RAPID-MIX API for an end-user applications and using its IML workflow, keeps the gulf of execution and evaluation short and the action-perception cycle fast, in both STU contexts for both developers and their end users. The design of the RAPID-MIX API can be understood as one which reduces the viscosity (Green, 1989) of ML developer tools. The RAPID-MIX API supported users engaging in and opportunistic approaches and active trial-and-error exploration of different solutions.

I have identified principles from other research contexts that bode well for the simplicity and minimalism underlying the RAPID-MIX API design. For instance, such principles relate with development tools that support creative thinking—“Make It As Simple As Possible - and Maybe Even Simpler” (Resnick et al., 2005)—or with low threshold (Myers et al., 2000) to make it easy for novices to get started with. In the perspective of adoption of innovations (Rogers, 2003), as a developer tool which is made more understandable, more comfortable to experiment with, and quicker at providing visible results, it is prone to be more quickly adopted. The RAPID-MIX API aligned, therefore, with principles related to rapid prototyping and making software design and development more efficient, accelerating the exploration of more alternatives, lowering the skills barriers, encouraging adoption, etc. This converges with von Hippel’s in User Innovation theory and the purpose of UITs for propelling user innovation. Elements of a toolkit such as the RAPID-MIX API should be easy to learn and with minimal training needs.

The principles that influenced the design of RAPID-MIX API as a ML developer toolkit for reducing the viscosity and skills barriers to IML systems development can be traced back to different origins. They have emerged organically from RAPID-MIX API designers’ experience and common sense, including prior user-centred research with Wekinator (Fiebrink et al., 2011), and insights and learned lessons through UCD research in RAPID-MIX. These principles can also be traced to theory that ingrains the RAPID-MIX API as design artefact (Sein et al., 2011), such as general principles from HCI (Carroll and Kellogg, 1989; Green, 1989) and creative technology (Resnick et al., 2005), and from management and innovation (Rogers, 2003; von Hippel, 2005).

7.4 Action Research and User-Centred Design in rapid innovation actions

This section discusses the limitations and quality criteria of my research, with regards to the application of Action Research in the context of RAPID-MIX. It also provides a reflection on the application of user-centred design and development to infrastructural software for rapid prototyping with IML, and set of recommendations based on learning obtained in the application of this methodology.

7.4.1 Limitations of research and quality criteria

As formulated in Section 5.2.2, my research was motivated by a practical problem in the context of RAPID-MIX, the application of a user-centred design process to the creation of a new user-friendly developer toolkit comprised of infrastructural software for the application of IML. RAPID-MIX became the substrate for investigating how faster and broader adoption of ML could be facilitated by designing ML developer tools with primary concerns for improved usability.

Action Research provided a framework that enabled me to intervene within the research client project setting and implement practical change with regards to the application of UCD. This provided for the need and concerns established by the research client at the beginning of the project—i.e. having a weak foundation in UCD and a potential lack of user engagement (Section 4.1). This framework guided me to find the adequate theoretical and methodological support in order to obtain valid contributions to knowledge.

Nevertheless, it is important to acknowledge that my research it limited to one organisational setting and one technology. My research also has limitations that derived from the chosen methodology, which I tried to minimise using quality criteria suggested by Davison et al. (2004). My research framed the RAPID-MIX API as the unit of analysis, there were two iterations in the study, and theoretical grounding was on von Hippel's User Innovation theory and HCML design spaces.

In Sections 4.5 and 5.9, I discussed the limitations inherent to the studies conducted in each AR cycle. In particular there were important limitations which recurred in the validation of the studies. There were also limitations related to the lightweight approach of the studies deployed, concerning the accuracy, precision and reliability of the data collected. Mostly these limitations were due to the contextual conditions of the project, such as intensive schedule, lack of budget (both time and money), lack of willing collaborators to all stages of each study. Potential bias that must be accounted are related to these factors, to the overload of information and potential lack of meaning. Some of the adopted mitigation strategies included making all the artefacts and recordings available to the consortium and other researchers, in an online, centralised repository, and triangulation of different data sources. Other strategies adopted to counteract the lack of formal and direct validation was to obtain indirect validation through the production of reports and deliverables.

Additional strategies were adopted for mitigating the typical methodological limitations of AR. I implemented quality criteria meant to improve the research quality at the macro level of the project. As previously identified in Section 3.2, the methodological limitations associated with Action Research can be mitigated by adhering to the principles and evaluation criteria of Canonical Action Research (Davison et al., 2004). As recommended by the AR methodology (Davison et al., 2004), I review how principles of CAR enabled me to strengthen my research design and mitigate the limitations of my chosen methodology and its application in the the project setting.

7.4.1.1 Research/Client agreement

The Research/Client Agreement (RCA) is the first evaluation criterion proposed by Canonical Action Research (Davison et al., 2004). The RCA for collaborative research within the RAPID-MIX consortium took place informally at the kick-off meeting at UPF, Barcelona, on February 2nd, 2015. In section 1.2.4, I provide more details about my entry to the research client system, role and responsibilities.

7.4.1.2 Cyclical process model

As reviewed in section 2.4.1, the Action Research cyclical model (Susman and Evered, 1978) proposed five stages: diagnosis, action planning, action taking, evaluation and specification of learning. CAR introduces the Cyclical Process Model (CPM), which subsumes the canonical AR cyclical model and proposes for research to be conducted in a sequential and systematic progression, with multiple iterations, in order to ensure for systematic rigour, reliability and transferability (Davison et al., 2004)

The UCD actions framework (Section 4.2) proposed three AR cycles that cut across the RAPID-MIX project life cycle. There was a deviation to the plan and the third cycle was deprecated. Each of the remaining two cycles, which I address in Chapters 4 and 5, contains a CPM instantiation which comprises diagnosis, action planning, action taking, evaluation, and reflection and knowledge specification.

7.4.1.2.1 Diagnosis

As Sein et al. (2011) suggest, the diagnosis stage is “needed to identify stakeholders, analyse problems and develop shared understandings and agreement about ’the’ problem(s) to be solved” (p. 10). In both AR cycles, I have engaged in collaborative discussion with other RAPID-MIX team members to reach an agreement about research priorities and needs. I also performed an independent diagnosis based on a reflection about elements of practice or aspects in RAPID-MIX which were considered to be in need of investigation. Each of the empirical chapters provides a summary of the diagnosis for the corresponding AR cycle.

7.4.1.2.2 Action planning

After the diagnosis stage, in which problems or research needs were identified, the action planning stage involved considering alternative actions in response. One of the contributions of this thesis was the UCD Actions framework, which sought to provide for the necessary flexibility to inform research needs that emerged as the RAPID-MIX project developed.

Section 4.2 introduced a general plan that I produced in collaboration with other Goldsmiths team members. This longitudinal plan set three action research cycles which covered all of the RAPID-MIX life cycle. This plan, which had more of a provisional and general nature due to the early stage at which it was produced, aligned the framework with the research that the project's long-term goals required.

In Chapters 4 and 5, I detailed the UCD actions of each cycle, and the specific needs that emerged as each AR cycle developed. In both AR cycles, for each of the interventions that took place, action planning was explicitly informed by the long-term goals that had been set on the longitudinal plan, and by the diagnosis stage, based on the insights that resulted from a previous cycle or previous intervention. I considered, proposed and discussed alternative courses of action, in terms of interventions that should follow, and how they could most effectively inform design decisions or evaluation of the RAPID-MIX API.

7.4.1.2.3 Action taking

The interventions have been deployed throughout both action research cycles, as part of the activities of WP2 documented Chapters 4 and 5. In the first action research cycle, I collaboratively engaged in a set of early UCD actions with RAPID-MIX stakeholders in a co-design workshop. I also engaged potential end-users for gathering user needs using background technologies as technological probes in a major hackathon. In our second action research cycle, following the design and development of the first version of the RAPID-MIX API, I carried out an expert elicitation and evaluation of a first generation prototype. I also deployed UCD actions for lightweight formative evaluation of different subsets of RAPID-MIX API with different user groups, including a general "broad-brush" usability evaluation with the Cognitive Dimensions questionnaire.

7.4.1.2.4 Evaluation

The evaluation stage of the CPM involved the analysis of the outcomes of the selected course of action. As mentioned in the previous subsection, we implemented different UCD actions throughout both action research cycles. The outcomes of these interventions have been evaluated internally and reported in periodic deliverables to the European Commission, and made available publicly. In the empirical Chapters 4 and 5, I present more expanded versions of the evaluation of UCD actions.

7.4.1.2.5 Reflection

The final stage of the CPM, which refers to the specification of emergent learning, is presented in the final sections of the chapters related to each AR cycle (Chapters 4 and 5, after a detailed treatment of the research interventions conducted. There, I reflect on the outcomes of the UCD actions and how they prepared the ground for the following AR cycle or the conclusion of the whole process. Furthermore, these outcomes also led to academic publications (Bernardo et al., 2017a, 2018). In Chapter 7, I distilled and formalised the learning from the overall process into generalised outcomes; I generalised the problem and the solution, deriving design principles from the research outcomes.

7.4.1.3 Theoretical grounding

An essential aspect to Action Research that must be considered to comply with the criteria of Canonical Action Research (Davison et al., 2004) is the grounding of action research in theory. Chapter 5.2 introduced the theoretical foundations of my research using User Innovation theory (von Hippel, 2005) and HCML design abstractions. Chapter 6 discusses how this grounding developed.

With the collaboration of colleagues at Goldsmiths, I devised a provisional model to evaluate and compare alternative designs. This stage, which corresponds to the initial period in which the "theory-free action learning" succeded and the first AR cycle. This model, which pursued more of a grounded theory development approach, focused on the relationships and interactions between stakeholders and technologies. This model has been published internally as a white paper, to provide a conceptual and reflection tool to inform the consortium.

Eventually, as my theoretical exploration developed and become fruitful through the discovery of User Innovation theory and the HCML design abstractions, I designed concepts maps—the User Innovation theory 7.1, and the EUI-ML and C-IML concepts maps (Figure 7.2 and Figure 7.3 respectively) to reframe and assemble these theories into a composite and unified framework (see Section 5.2.3 for details about theoretical frame) for reflection, sensemaking and knowledge abstraction.

7.4.1.4 Change through action

My involvement in the planning, design and deployment interventions was motivated by the goal to contribute to the problem that the RAPID-MIX consortium pursued—how to design and evaluate the RAPID-MIX API as a toolkit for innovation with IML. The interventions addressed the design of usable, useful and accessible tools for interactive machine learning. The interventions that have been deployed yielded good results and informed positive change within the project, which will be supported by each of the following empirical chapters,

My contributions to the project spanned many different areas within the project. Most importantly, I was able to engage in collaborative research, which provided fruitful results. In particular, my contributions to improving a specific capacity within the RAPID-MIX consortium: applying user-centred design consistently using the UCD actions framework, which has been adopted, employed independently and successfully by different partners within the consortium. The RAPID-MIX consortium approved the plan for interventions, their scheduling and deployment, and reports with the resulting outcomes.

These contribution, to the project setting and research client have been documented in official deliverables. They were sent for official review to the European Commission, where they were accepted. This shows the motivation and effort of the RAPID-MIX consortium towards adopting and recognising the actions for positive change.

7.4.1.5 Learning through reflection

According to general AR principles (Kock, 2007), a model is needed to capture the reality of the project and abstract it for analysis, reflection and knowledge specification. A linear model for innovation, such as the typically closed innovation pipeline, would fail to capture the complexity of the process that took place in RAPID-MIX.

As discussed in Chapter 7, I have developed concepts maps—the EUI-ML and C-IML concepts maps (Figure 7.2 and Figure 7.3 respectively) to reframe and assemble the contributing theoretical bases into a composite model and unifying framework (see Section 5.2.3 for details about theoretical framing). I employ this model for reflection, sensemaking and knowledge distillation. I used it to discuss and interpret the insights resulting from UCD actions, and to articulate this insights with theory to formalise the knowledge that emerged in my research.

7.4.2 Recommendations for Action Research and User-Centred Design in rapid innovation actions

In RAPID-MIX, we had to implement UCD on a large scale and time frame. Overall, we found that UCD actions for lightweight evaluation provided gains that compared favourably to their operational costs. Using a lightweight approach, we managed to assess a broad set of related artefacts (i.e. RAPID-MIX API subsets, documentation and examples) iteratively in the wild. This approach seems particularly well-suited for a technology with the level of indirection, and dependency on documentation and examples, such as the RAPID-MIX API or other ML software development toolkits. UCD actions, despite being lightweight, still require a great deal of thought and preparation to be engaging and simultaneously useful to participants and organisers. The data organisation and analysis can require significant time and effort, which is difficult to estimate before the action.

In the overall UCD cycle, the outcomes of one stage must be clear and of consequence to the next stage (i.e. insights obtained from one UCD action should inform a subsequent design or development stage, and the design stage outcome should be used for inquiry in the next UCD action). This requires sequencing and integration with the overall development cycle, which may or may not be straightforward. Here, knowledge of UCD methodology alone was insufficient; management skills and practical working experience with UCD were invaluable, as was intuition to strike the right balance between breadth and depth of assessment.

Based on the practical experience of our UCD process implementation, I distilled a set of recommendations for best practices for UCD in rapid innovation actions. They are as follows:

1. *UCD champion* – first of all, projects adopting UCD methodologies will benefit from having a UCD champion, someone who can make use of their empowered role in an organisation—or who is formally empowered for such a role—and whose presence and influence remains constant throughout the process. The role of this actor will contribute effectively to 1) help disseminate the UCD ethos across the different stakeholders of the project, 2) engage the project's key stakeholders with the different actions and outcomes, and 3) manage the iterative cycles of the UCD process and the system-centric backlash, ensuring that UCD actions are collaboratively deployed and that the resulting knowledge flows across the organisation to inform design.

2. *Begin the process with promoting a shared vision and a UCD ethos with key stakeholders* – begin the UCD process by promoting a shared vision and by aligning the key stakeholders' thoughts and understanding about the design opportunity. Engage with multiple stakeholders early on and continuously using co-design workshops to unlock tacit knowledge and validate research opportunities (e.g. Section 4.3.5.1). Use design exercises such brainstorming and ideation, and design artefacts and techniques (e.g. post-it notes, storyboards, scenarios, personas) to capture and embody explicit and tacit knowledge as much as possible. Make the outcomes visible and turn them into devices to foster discussion among stakeholders. As previously discussed in Section 4.3.5, this approach with key stakeholders yielded good results at the beginning of the project. We also observed that new collaborators arriving at later stages to the project—particularly ones with crucial roles—skipped a more profound or more experiential introduction to the methodology. This had implications in terms of organisational backlash to the UCD methodology, as discussed in Section 6.5.

3. *Scope and understand the user* – by scoping and understanding the user, we want to optimise our overall design process by grounding it among the most precise user population as possi-

ble; the goal is to delimit possibilities and constrain effort (Ritter et al., 2014). We want to understand the needs, goals and values, tasks and context-of-use of this specific user target population. This process can take approaches from different domains (e.g. Human factors, psychology, ethnography, design/art) and with different levels of analysis (e.g. anthropometric, perceptual, cognitive, social). What techniques should then be applied? There is a considerable body of techniques available for UCD practitioners to apply for the derivation of design insights. When we designed our UCD methodology, we surveyed and selected a set of techniques from an extensive body of academic papers and design sources (see Section 2.2.2), and prescribed criteria for their application in our UCD actions (e.g. purpose of the study, target user group, data output type, cost and project stage) (Appendix B). These criteria for selecting techniques for user engagement and data analysis can be useful. Additionally, one should consider local context factors such as the project's scale, the team members involved, the target technologies and the specific design goals for each case.

4. *Scope the design space* – this process typically begins with a speculative space of design, populated with ideas and potential concepts, and learned lessons from previous experiences, based on previous assumptions. The design space evolves gradually and gets populated with prototypes that provide different validations, and becomes constrained by guidelines and specifications. Eventually, the design space becomes populated with real designs and groups of designs that characterise it. Throughout the project, our UCD actions enabled us to scope the design space of RAPID-MIX technologies gradually (Sections 4.4, 1.2.2, 5.2.4 and 5.7). We began by capturing seed ideas from stakeholders in co-design sessions, starting with a set of background technologies. We used these as technological probes for the needs and goals of the set of potential users with early UCD actions, such as a large scale hackathon and user engagement at maker fairs. Such interventions contributed to an initial scoping of the design space, supported the definition of prototyping guidelines and API design guidelines. Prototypes identified features, new concepts, technological integrations, i.e. plausible designs that demonstrated the possibilities of the RAPID-MIX design space, and simultaneously matched user needs. The subsequent prototyping iterations also informed the design of the API, the examples and demonstrators that were, in turn, validated with users.

5. *Direct the UCD actions and prototyping activities using research questions* – use specific research questions to lead the design of user engagement interventions and prototypes. Obviously, the same questions should ground the analysis of the data that results from the interventions. As advised by the UCD actions framework (Section 4.2.1), in RAPID-MIX, we followed design traditions that use interventions and artefacts as embodiments of research questions and design knowledge (e.g. Sections 4.1, 4.3.3).

6. *Organise your data for shared consultation* – the amount of generated data can vary greatly depending on the scale of the design project. For instance, hundreds of post-it notes, dozens of videos from interviews, can be generated in a few hours of a workshop, depending on the

number of participants and team members involved in the action. With online collaborative work tools available (e.g. Google drive, Youtube, Gitlab, etc.), it is not only possible but also desirable to store and organise data so that multiple stakeholders can access it. This allows multiple cuts through data by different stakeholders and leads to different insights. In RAPID-MIX, the interviews that resulted from specific UCD actions (i.e., hackathons and workshop participant interviews, co-design and heuristic evaluation with project stakeholders, etc.) have been video recorded, stored and annotated on a private YouTube channel. It is also important to think of the specific stakeholders and format that is used to communicate the insights, such as design workbooks for the general audience, or issues in code repositories for the development team.

7. *Build models to make sense of data* – engaging with users in need-finding can generate an extensive amount of data and this can turn sensemaking into an overwhelming activity. All the data points and pieces of information (e.g. post-its, scenarios, storyboards, interviews, prototypes) need to be organised and abstracted for synthesis of design insights. Using analysis techniques for primary data analysis, such as design personae, diagrams or external theories can help build from simple to more complex models. Ultimately, we want to support higher-level perspectives and inform design decisions. In RAPID-MIX, I have created simple models for analysing the actors space, seed ideas and prototypes, through to more complex models based on external theories and frameworks, such as the User Innovation theory (von Hippel, 2005) (Section 7.2) and the design spaces for interaction with machine learning (Amershi, 2012; Dudley and Kristensson, 2018). These models have been disseminated internally, for project stakeholders to use. They were also released publicly for the research community and general audience as an academic publication. We obtained positive feedback about how they provide an effective tool for understanding.

8. *Manage the UCD process and its ROI* – the effort of scoping and understanding the user can become an endless journey. There should be careful consideration about specific user typologies, tasks and contexts of use, and about the technologies involved in the design opportunity. UCD has been considered successful mostly when applied to the design of physical artefacts and graphical user interfaces. However, there are technologies to which the application of the process may not be straightforward, and it might become difficult to verify success or recognise benefits. In RAPID-MIX, as discussed in Section 6.5, the substantial efforts invested in applying UCD to designing the machine learning API received internal remarks about its limited payoff. In order to overcome these issues, the outcomes of one stage must be clear and of consequence to the next stage (i.e. the outcomes of one UCD action should inform a subsequent design stage, and the following UCD action should use the outcomes of the design stage). While it may appear evident, there are circumstances where conflicting goals (e.g. dissemination and marketing events) can undermine the process.

Chapter 8

Conclusion

Approximately four years ago, as RAPID-MIX started, the strategic goals and directions that were initially laid out for the project, pursued what was then an under-explored niche in research and development—improving the competitiveness of SMEs in the creative industries sector with new machine learning tools. Four years past, these goals and the path taken to accomplish them revealed the vision and foresight of the group of researchers that bootstrapped RAPID-MIX. This vision, which has been growing otherwise in an analogous and independent fashion in the software development mainstream, refers to the importance of designing tools to push forward innovation with machine learning, and democratising the access to the potentials of this technology to creative start-ups and individuals.

Today, the hype around how machine learning is and will continue to impact the next wave of innovation is greater than ever before. "Gold rush" and "feeding frenzy" are used to characterise this feedback loop, which attracts the attention and desire of individuals, organisations, and even governments, at a national level. There is a generalised interest in profiting from this opportunity and getting ahead at the competitive level, by mobilising efforts of the globally increasing population of developers towards new economic efficiencies powered by machine learning. In this process, the unique challenges that ML presents to developers (Patel et al., 2010) need to be superseded or attenuated. ML developer tools such as application programming interfaces (APIs) pose the first barrier and should be, therefore, better understood. The design of such tools presents distinct challenges from other design processes in which user-centred methodologies are often used (e.g. the design of end-user-facing graphical interfaces) (Myers and Stylos, 2016).

During the development of RAPID-MIX (2015–2018), many important open-source ML frameworks and software toolkits appeared in the ML tooling landscape[1]. Some of the most popular ML developer tools are backed by major technological conglomerates—e.g. Tensorflow (Google), Pytorch (Facebook), CNTK (Microsoft), MXNet (Amazon). Other ML toolkits are open-source projects from individuals which provide interesting examples of user innovation. Some of these ML

[1] Deep Learning Framework Power Scores 2018 (empirically-sourced evidence) https://www.kaggle.com/discdiver/deep-learning-framework-power-scores-2018

toolkits have been acknowledgely designed with usability as a primary concern. Keras (Chollet, 2015), for instance, is an open-source deep learning API which, according to Chollet, was designed for usability and was based on usability principles—consistent and simple APIs, minimal number of user actions required for common use cases, and clear and actionable feedback upon user error. The usability-focused innovation in ML API design of Keras has led to its recent adoption as one of the main interfaces of Tensorflow.

At this juncture, the context has become ripe with the opportunity to make machine learning more accessible to new groups of people. Particularly, it is now very desirable and relevant to have people who are not ML experts but rather domain experts applying machine learning for problem-solving in various domains and in their own design problems. In a world that is more and more data-driven and where the global population of software developers is growing, machine learning is becoming a tool that is not so much out of reach anymore, and for which it will be essential to have a minimum amount of literacy and ability to use.

Until recently ML was often considered difficult by those who did not have a computer science or a more quantitative background such ML experts or data scientists. However, besides the RAPID-MIX API, there have been other recent initiatives following up on this trend and focusing on the development of tools to democratise ML—e.g. ESP (Mellis et al., 2017), TuriCreate, ML5js.

8.1 Summary of Thesis Achievements and Contributions

My thesis contributes with new understanding to the application of a user-centric approach to explore and better understand how machine learning developer tools may be designed to facilitate learning, use and rapid adoption by creative developers. This research is especially important to understand human-centred approaches to the design of ML infrastructural software and APIs, and how they can build upon a stronger and more nuanced connection between ML API designers and end-users.

The following sections outline the main research outcomes concerning the application of User-Centred Design in RAPID-MIX and the main achievements in the design, development and evaluation of the RAPID-MIX API.

8.1.1 UCD Actions methods framework

My research contributes a methods framework of *UCD Actions* for collaborative design and evaluation of infrastructural software and developer tools for rapid prototyping with interactive machine learning. The *UCD Actions* framework enabled us to think, plan and deploy, deliberately and collaboratively, different user interventions, including workshops, hackathons, user feedback sessions, and a usability study with an adapted version of the Cognitive Dimensions questionnaire. These

UCD actions allowed us to gain a deeper understanding of the targeted user groups—i.e. hackers, makers, professional audio software developers and creative coders—and of their contexts.

Early UCD actions were deployed collaboratively to inform about stakeholders and end-users and help the formulation of design guidelines for the RAPID-MIX API. Later UCD actions for lightweight evaluation of the RAPID-MIX API were particularly useful for the assessment of a more comprehensive set of elements of the toolkit (C++ and JS APIs, documentation and code examples) with different user groups "in the wild". With these interventions, we uncovered conceptual and technical limitations and informed subsequent improvements to the ML developer toolkit and its documentation.

The *UCD actions* framework enabled us to choose adequate methods to engage users and stakeholders to obtain feedback about how ML developer tools affect their developer experience and to explore usability issues, design decisions and trade-offs of developer tools for rapid prototyping with IML. This framework was designed for reuse and provides protocol which can be most useful for application in the context of other rapid innovation actions such as RAPID-MIX.

8.1.2 Consequences and implications of a UCD approach to ML developer tools

In general, a UCD approach helped us to recognise user needs and expectations for simple and effective ML developer tools. A UCD approach helped us to design ML developer tools that were usable, useful, and sufficiently robust for differently skilled creative developers, with little to no ML experience, in different scenarios and with different use cases. We observed how participants in UCD actions were able to quickly grasp and successfully use different subsets of the RAPID-MIX API. Many participants were able to produce fully working hacks and prototypes, which contributed to understanding and further refining the scope of the design space of the RAPID-MIX API. We were able to find usability problems and directions of improvement, mostly related to documentation fragmentation, lack of support for understanding intricate working details, error support and recoverability, and lack of evaluation guidance.

Overall, we found that ML developer tools, such as the RAPID-MIX API, can be designed for user friendliness and ease of use, and become enabling and empowering toolkits for creative ML-non-expert developers applying IML for rapid prototyping of multimedia applications. Nevertheless, the design trade-offs of ML tools such as the RAPID-MIX API give them a limited scope of application. Mostly, the RAPID-MIX API was considered to be adequate for opportunistic approaches to rapid prototyping, proofs-of-concept and simple end-user applications in commercial development of creative technology. User-friendly ML tools can be most useful for teaching introductory ML courses and ML design patterns to ML-non-expert developers. Moreover, they can be useful to individuals and communities in art, design, and multimedia, that aim to explore and integrate IML workflows into their creations.

8.1.3 Interactive machine learning for user-innovation toolkits framework

My research contributes a theoretical framework for human-centred and holistic analysis and understanding of the design factors, constraints and features of user-friendly ML developer tools for creative ML-non-expert lead users. This framework abstracts and articulates von Hippel's (2005) User Innovation theory with HCML design abstractions, namely Amershi's (2012) and Dudley and Kristensson's (2018) design spaces, contributes cognitive tools such as conceptual maps with the taxonomy of User Innovation theory, EUI-ML and C-IML design spaces (Figures 7.1, 7.2 and 7.3, respectively).

I provided a brief illustration of how this framework can provide a working theory for exploring, informing and analysing the design of toolkits for end-user innovation with IML. I argued that such a framework can help us understand better the design space of ML developer tools, such as the RAPID-MIX API. Moreover, such a framework can support sensemaking in the demanding contexts and multiple layers of collaborative, multi-institutional and rapid innovation actions and technology transfer such as the EU H2020-funded RAPID-MIX.

8.1.4 Design principles of user-centred infrastructural software for IML

My action research contributes with new understanding about the process of designing user-centred developer toolkits for rapid prototyping with IML. This includes an empirical perspective which resulted in the identification and distillation of principles that guided the collaborative design and development of the RAPID-MIX API. These principles comprise 1) providing real-time interactivity between user, data and ML algorithms, 2) scaling IML systems development with user-centred infrastructural software, and 3) reducing the skills and usage barriers to IML systems development.

These principles were influenced by the understanding gathered as an outcome of application of the ADR methodology and as part of the UCD approach to inform the creation and refinement of ML infrastructural software. In particular, they were influenced by understanding about the needs, goals, expectations of ML-non-expert developers as users of ML infrastructural software, their experience developing and rapid prototyping with RAPID-MIX API and the IML workflow, and the multiple contexts around these users and their end-users. Additionally, I complemented this perspective with a theoretical discussion supported by User Innovation theory and HCML design abstractions, which added a useful level of examination interpretation.

8.1.5 Recommendations for applying UCD in innovation actions

The user-centred interventions that were planned and deployed in RAPID-MIX and their outcomes, illustrate the effectiveness of our collaborative UCD approach and of our UCD actions framework. UCD actions in RAPID-MIX contributed with useful iterations to the overall design process of our toolkit. The practical experience of applying multiple UCD iterations in a multi-stakeholder innovation project comes with a significant caveat: even lightweight UCD actions require careful management of the potential organisational backlash, and of the significant effort required to achieve recognisable effectiveness and ROI. With the knowledge and learning through action obtained through empirical experience, I provide practical insights which may be useful for longitudinal studies of rapid innovation actions such as RAPID-MIX.

8.1.6 Practical contributions to the design and development of the artefact

As described in Section 1.2.4, I contributed extensively to the overall design process, which includes activities in the different work packages. I also contributed specifically to the design and development of the RAPID-MIX API and associated artefacts. In Section 5.6.3, I describe the software development contributions that I made to the main code base and to the API demonstrators. In Section 5.7.5.1, I describe the empirically-derived and formal recommendations that I made to improve the design of RAPID-MIX API.

8.2 Future Work

Research in areas that intersect ML and design is growing, including rapid prototyping, design education, patterns and improved documentation techniques. These areas comprise some of the most important directions for design research around ML. Specifically, in my vision of future research in the design of ML developer tools, I would be interested focusing on the following avenues:

a) Further investigating how to build upon the experiential learning and shortened action-perception loop of IML as a design pattern for the improvement of education, adoption and innovation with ML. This includes investigating new and more effective metaphors, richer forms of documentation and multimedia, techniques for structuring advanced organisers, for improving pedagogical approaches to the probabilistic and non-deterministic nature of ML.

b) Investigating better ways and the implications of integrating the IML workflow in distinct domain-specific user-innovation toolkits. This includes investigating better ways to integrate and segregate domain-specific requirement as part of the working framework of developers and as external to the notations structure.

c) Deepening the investigation about new design methods and evaluation techniques for Human-centred Machine learning, and adjusting established methods and techniques to take into account the opaqueness and non-deterministic nature of ML. This includes two main avenues of research which explore the CDs as pragmatic methodological tools for ML API and notation designers:

- The first avenue is to investigate a more extended and formalisable set of dimensions, which may help to analyse ML and ML APIs more adequately. New dimensions have been formulated in previous research, as well as criteria for formalising new dimensions (Blackwell, 2000). New candidate dimensions could facilitate, for instance, the analysis of data and ML data sets as part or outsiders to the *working framework* and the *work-step unit*. There are also opportunities for conceptualising and formalising new dimensions related to the non-deterministic or stochastic nature of ML algorithms.
- The second avenue of research is to explore ways to augment the CDs framework to account more holistically for a set of interdependent artefacts—such as language notations, programming interfaces, development environments and documentation—and other high-level aspects which Petre (2006) has identified. According to Petre, scope, trade-offs, processes, culture, convention, and the context of use, should all be accounted. She also proposed that this could be achieved through a hybrid technique which integrates facets theory with the CDs, for a more complete and holistic analysis.

Bibliography

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Jon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Oriol Vinyals, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2015.

Ableton. Ableton Live. URL `https://www.ableton.com/`.

Chadia Abras, Diane Maloney-Krichmar, and Jenny Preece. User-Centered Design. In W. Bainbridge, editor, *Encyclopedia of Human-Computer Interaction*, volume 37, pages 445–56. Thousand Oaks: Sage Publications, 2004. ISBN 0974309125. doi: 10.1.1.94.381. URL `http://uba-mobile.googlecode.com/svn/trunk/resources/Abras,Maloney-krichmar,Preece/Bainbridge,W.EncyclopediaofHuman-ComputerInteraction.ThousandOaksSagePublications-Abras,Maloney-Krichmar,Preece-2004-User-CenteredDesign.pdf`.

Zoltan J. Acs, David B. Audretsch, and Maryann P. Feldman. R&D Spillovers and Recipient Firm Size. *The Review of Economics and Statistics*, 76(2):336–340, 1994.

Anne Adams and Anna L. Cox. Questionnaires, in-depth interviews and focus groups. In *Research Methods for Human Computer Interaction*, pages 17–34. 2008. ISBN 9780521870122.

All European Academies. The European Code of Conduct for Research Integrity, 2017.

Saleema Amershi. *Designing for Effective End-User Interaction with Machine Learning.* PhD thesis, University of Washington, 2012.

Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine*, 35(4):105–120, 2014. ISSN 0738-4602. doi: 10.1609/aimag.v35i4.2513. URL `http://aaaipress.org/ojs/index.php/aimagazine/article/view/2513`.

Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, pages 337–346, 2015. doi: 10.1145/2702123.2702509. URL `http://dl.acm.org/citation.cfm?doid=2702123.2702509`.

Chad A Austin. *Renaissance: a functional shading language*. PhD thesis, Iowa State University, 2005.

David Avison, Richard Baskerville, and Michael Myers. Controlling action research projects. *Information Technology & People*, 14(1):28–45, 2001. ISSN 09593845. doi: 10.1108/09593840110384762.

R. Baskerville and AT Wood-Harper. Diversity in information systems action research methods. *European Journal of ...*, 7(2):90–107, 1998. ISSN 0960-085X. doi: 10.1057/palgrave.ejis.3000298. URL `http://www.ingentaconnect.com/content/pal/0960085x/1998/00000007/00000002/3000298`.

Richard L. Baskerville. Investigating information systems with action research. *Communications of AIS*, 2(3):4, 1999. ISSN 1529-3181. doi: http://www.cis.gsu.edu/~rbaskerv/CAIS_2_19/CAIS_2_19.html. URL `http://portal.acm.org/citation.cfm?id=374476`.

Matt Bellingham, Simon Holland, and Paul Mulholland. A cognitive dimensions analysis of interaction design for algorithmic composition software. In *Proceedings of Psychology of Programming Interest Group Annual Conference*, pages 135–140, 2014.

Francisco Bernardo, Atau Tanaka, Rebecca Fiebrink, Adam Parkinson, Sebastian Meala, and Frédéric Bevilacqua. D2.1 User-Centred Design Methodology. pages 1–36, 2015. URL `https://rapidmix.goldsmithsdigital.com/wp-content/uploads/2016/02/D2.1-User-Centred-Design-Methodology1.pdf`.

Francisco Bernardo, Nicholas Arner, and Paul Batchelor. O Soli Mio: Exploring Millimeter Wave Radar for Musical Interaction. *NIME 2017 Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 283–286, 2017a.

Francisco Bernardo, Rebecca Fiebrink, Frédéric Bevilacqua, Sebastián Mealla, and Michael Zbyszynski. D2.5 Final Design Specification. Technical report, 2017b.

Francisco Bernardo, Mick Grierson, and Rebecca Fiebrink. User-Centred Design Actions for Lightweight Evaluation of an Interactive Machine Learning Toolkit. *Journal of Science and Technology of the Arts*, 10(2):2, 2018. ISSN 2183-0088. doi: 10.7559/citarj.v10i2.509. URL `http://artes.ucp.pt/citarj/article/view/509`.

Frédéric Bevilacqua, Rémy Müller, and Norbert Schnell. MnM: a Max/MSP mapping toolbox. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 85–88. National University of Singapore, 2005.

Frédéric Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, Fabrice Guédy, and Nicolas Rasamimanana. Continuous Realtime Gesture Following and Recognition. pages 73–84, 2010. URL http://articles.ircam.fr/textes/Bevilacqua09b/index.pdf.

Fréderic Bevilacqua, Francisco Bernardo, Sebastián Mealla, and Rebecca Fiebrink. D2.2 Design Guidelines for Prototyping. Technical report, RAPID-MIX, 2015. URL http://rapidmix.goldsmithsdigital.com/wp-content/uploads/2016/02/D2.2DesignGuidelineforPrototyping.pdf.

Mark Bilandzic and John Venable. Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics. *The Journal of Community Informatics*, 7(3):1–17, 2011. ISSN 1712-4441. URL http://ci-journal.net/index.php/ciej/article/view/786/804.

Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. 2006. ISBN 9780387310732. doi: 10.1117/1.2819119. URL http://www.library.wisc.edu/selectedtocs/bg0137.pdf.

Alan F. Blackwell. Dealing with New Cognitive Dimensions. *Workshop on Cognitive Dimensions*, (1986), 2000. doi: 10.1.1.18.7947. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.7947.

Alan F Blackwell. First Steps in Programming: A Rationale for Attention Investment Models. In *Proceedings - IEEE 2002 Symposia on Human Centric Computing Languages and Environments, HCC 2002*, pages 2–10, 2002. ISBN 0769516440. doi: 10.1109/HCC.2002.1046334.

Alan F Blackwell and Thomas R G Green. A Cognitive Dimensions Questionnaire Optimised for Users. In *Proceedings of 12th Workshop of the Psychology of Programming Interest Group (PPiG)*, number April, pages 137–154, 2000. URL http://www.ppig.org/papers/12th-blackwell.pdf.

Alan F. Blackwell and Thomas R. G. Green. Notational systems – the cognitive dimensions of notations framework. *{HCI} Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, pages 103+, 2003. doi: 10.1016/S0304-3975(01)00143-8. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.2023.

Pontus Braunerhjelm, Ding Ding, and Per Thulin. The knowledge spillover theory of intrapreneurship. *Small Business Economics*, 51(1):1–30, 2017. ISSN 15730913. doi: 10.1007/s11187-017-9928-9.

Gerard Briscoe and Catherine Mulligan. Digital Innovation: The Hackathon Phenomenon. 2014.

Tim Brown. *Change by Design: How Design Thinking Transforms Organisations and Inspires Innovation.* HarperCollins Publishers, New York, 2009. ISBN 9788578110796. doi: 10.1017/CBO9781107415324.004. URL https://next.ft.com/

content/590c0dd2-7801-11e5-933d-efcdc3c11c89?ftcamp=crm/email/20151022/nbe/CommodsDaily/product.

Lars Buitinck, Gilles Louppe, and Mathieu Blondel. API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv: ...*, pages 1–15, 2013. URL http://arxiv.org/abs/1309.0238.

Jamie Bullock and Ali Momeni. ml.lib: Robust, Cross-platform, Open-source Machine Learning for Max and Pure Data. In *NIME 2015 Proceedings*, pages 3–8, 2015.

CJC Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Scandinavian Journal of Medicine and Science in Sports*, Data Minin(2):121–167, 1998. ISSN 16000838. doi: 10.1111/sms.12977.

Baptiste Caramiaux, Nicola Montecchio, Atau Tanaka, and Frédéric Bevilacqua. Adaptive Gesture Recognition with Variation Estimation for Interactive Systems. *ACM Transactions on Interactive Intelligent Systems*, 4(4):1–34, 2014. ISSN 2160-6455. doi: 10.1145/2643204.

Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The Design Space of Input Devices. In *CHI'90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, number April, pages 117–123, 1990.

J. M. Carroll and W. A. Kellogg. Artifact as theory-nexus: hermeneutics meets theory-based design. *ACM SIGCHI Bulletin*, 20(SI):7–14, 1989. ISSN 07366906. doi: 10.1145/67450.67452. URL http://portal.acm.org/citation.cfm?doid=67450.67452.

Erin Cherry and Celine Latulipe. Quantifying the Creativity Support of Digital Tools through the Creativity Support Index. *ACM Transactions on Computer-Human Interaction*, 21(4):1–25, 2014. ISSN 10730516. doi: 10.1145/2617588. URL http://dl.acm.org/citation.cfm?doid=2633907.2617588.

H. Chesbrough. Open Innovation: A New Paradigm for Understanding Innovation. In H. Chesbrough, W. Vanhaverbeke, and J. West, editors, *Open Innovation: Researching a New Paradigm*, page 373. Oxford University Press, New York, 2006.

Parmit K Chilana, Mary P Czerwinski, Tovi Grossman, Chris Harrison, Ranjitha Kumar, Tapan S Parikh, and Shumin Zhai. Technology Transfer of HCI Research Innovations: Challenges and Opportunities. *Extended Abstracts of the ACM CHI'15 Conference on Human Factors in Computing Systems*, 2:823–828, 2015a. doi: 10.1145/2702613.2724724. URL http://dx.doi.org/10.1145/2702613.2724724.

Parmit K Chilana, Andrew J Ko, and Jacob O Wobbrock. From User-Centered to Adoption-Centered Design: A Case Study of an HCI Research Innovation Becoming a Product. In *CHI 2015*, pages 1749–1758, 2015b. ISBN 9781450331456.

François Chollet. Keras. \url{https://keras.io}, 2015.

Sharon Lynn Chu, Francis Quek, Yao Wang, and Rex Hartson. Finding-NEVO: Toward radical design in HCI. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8117 LNCS(PART 1):471–478, 2013. ISSN 03029743. doi: 10.1007/978-3-642-40483-2_33.

Steven Clarke. What is an End User Software Engineer? In *End-User Software Engineering*, pages 1–2. 2007.

Steven Clarke. How Usable Are Your APIs? In *Making Software: What Really Works, and Why We Believe It*, number November, pages 545–565. O'Reilly Media, Inc., 2010.

Steven Clarke and Curtis Becker. Using the Cognitive Dimensions Framework to evaluate the usability of a class library. In *Proceedings of the First Joint Conference of EASE and PPIG*, number April, pages 359–366, 2003. URL http://www.ppig.org/papers/15th-clarke.pdf.

Wesley M. Cohen and Daniel A. Levinthal. Absorptive Capacity: A New Perspective on Learning and Innovation. *Administrative Science Quarterly*, 35(1):128–152, 1990. ISSN 0143-2095. doi: 10.1177/0149206310369939.

A. Cooper, R. Reimann, and D. Cronin. *About Face 3: The Essentials of Interaction Design.* Wiley, Indianapolis, Indiana, 2007.

Alan Cooper. *The Inmates Are Running the Asylum.* Sams Publishing, 1999. ISBN 0672316498. doi: 10.1007/978-3-322-99786-9_1.

Robert G Cooper. Stage-Gate Systems: A New Tool for Managing New Products. *Business Horizons*, 33(3):44–54, 1990. ISSN 00076813. doi: 10.1016/0007-6813(90)90040-I.

Nuno N Correia and Atau Tanaka. User-Centered Design of a Tool for Interactive Computer-Generated Audiovisuals. *ICLI*, 19, 2014. URL http://users.fba.up.pt/{~}mc/ICLI/correia.pdf.

Mary M. Crossan and Marina Apaydin. A multi-dimensional framework of organizational innovation: A systematic review of the literature. *Journal of Management Studies*, 47(6):1154–1191, 2010. ISSN 00222380. doi: 10.1111/j.1467-6486.2009.00880.x.

Krzysztof Cwalina and Brad Abrams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries.* Addison-Wesley, 2nd edition, 2009.

Jason Dagit, Joseph Lawrance, Christoph Neumann, Margaret Burnett, Ronald Metoyer, and Sam Adams. Using cognitive dimensions: Advice from the trenches. *Journal of Visual Languages and Computing*, 17(4):302–327, 2006. ISSN 1045926X. doi: 10.1016/j.jvlc.2006.04.006.

Robert M. Davison, Maris G. Martinsons, and Ned Kock. Principles of canonical action research. *Information Systems Journal*, 14(1):65–86, 2004. ISSN 13501917. doi: 10.1111/j.1365-2575.2004.00162.x.

A.K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu. a CAPpella: programming by demonstration of context-aware applications. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 6(1):40, 2004. ISSN 1581137028. doi: 10.1145/985692.985697. URL `http://portal.acm.org/citation.cfm?id=985697`.

James Diprose, Bruce MacDonald, John Hosking, and Beryl Plimmer. Designing an API at an appropriate abstraction level for programming social robot applications. *Journal of Visual Languages and Computing*, 39:22–40, 2017. ISSN 1045926X. doi: 10.1016/j.jvlc.2016.07.005. URL `http://dx.doi.org/10.1016/j.jvlc.2016.07.005`.

Alan Dix. Action Research in HCI, 2012. URL `https://alandix.com/blog/2012/12/10/action-research-in-hci/`.

Pedro Domingos. A Few Useful Things to Know about Machine Learning. *Communications of the ACM*, 55(10):78–87, 2012.

John J Dudley and Per Ola Kristensson. A Review of User Interface Design for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.*, 8(2):8:1—-8:37, jun 2018. ISSN 2160-6455. doi: 10.1145/3185517. URL `http://doi.acm.org/10.1145/3185517`.

Henry Edison and Richard Torkar. Towards innovation measurement in the software industry. *The Journal of Systems and Software*, 86:1390–1407, 2013.

W Keith Edwards, Victoria Bellotti, Anind K Dey, and Mark W Newman. Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Infrastructure. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '03*, pages 297–304, 2003. ISBN 1581136307. doi: 10.1145/642611.642664. URL `http://doi.acm.org/10.1145/642611.642664`.

Jerry Alan Fails and Dan R Olsen. Interactive Machine Learning. In *Proceedings of the 8th international conference on Intelligent user interfaces IUI 03*, pages 39–45, 2003. ISBN 1581135866. doi: 10.1145/604045.604056. URL `http://portal.acm.org/citation.cfm?doid=604045.604056`.

Jakub Fiala, Matthew Yee-King, and Mick Grierson. Collaborative coding interfaces on the Web. In *International Conference of Live Interfaces (ICLI 2016)*, pages 49–58, 2016. URL `http://users.sussex.ac.uk/{~}thm21/ICLI{_}proceedings/2016/Papers/Long{_}Papers/109{_}Collaborative{_}Coding{_}Interfaces{_}Web.pdf`.

Rebecca Fiebrink, Perry R Cook, and Daniel Trueman. Human Model Evaluation in Interactive Supervised Learning. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, page 147, 2011. ISBN 9781450302289. doi: 10.1145/1978942.1978965. URL `http://portal.acm.org/citation.cfm?doid=1978942.1978965`.

Rebecca Anne Fiebrink. *Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance.* PhD thesis, 2011a. URL `http://www.cs.princeton.edu/{~}fiebrink/drop/thesis/thesis{_}kbow{_}conclusions.pdf`.

Rebecca Anne Fiebrink. *Real-Time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance.* PhD thesis, Princeton University, 2011b. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.226.1409{&}rep=rep1{&}type=pdf.

Steven Flowers and Georgina Voss. User Innovation in the Music Software Industry: The Case of Sibelius. In Candace Jones, Mark Lorenzen, and Jonathan Sapsed, editors, *The Oxford Handbook of Creative Industries*. Oxford University Press, 2013. doi: 10.1093/oxfordhb/9780199603510.013.004.

Frederic Font, Gerard Roma, and Xavier Serra. Freesound Technical Demo. In *Proceedings of the 21st ACM international conference on Multimedia - MM '13*, pages 411–412, 2013. ISBN 9781450324045. doi: 10.1145/2502081.2502245. URL http://dl.acm.org/citation.cfm?doid=2502081.2502245.

Marcus Foth and Jeff Axup. Participatory design and action research: Identical twins or synergetic pair. *Participatory Design Conference PDC Trento Italy*, i:93–96, 2006. doi: 10.1007/s11213-009-9145-9. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.6342{&}rep=rep1{&}type=pdf.

Martin Fowler. Module Assembly. *IEEE SOFTWARE*, pages 65–67, 2004.

Jules Françoise, Norbert Schnell, and Frédéric Bevilacqua. A Multimodal Probabilistic Model for Gesture-based Control of Sound synthesis. In *Proceedings of the 21st ACM international conference on Multimedia - MM '13*, pages 705–708, 2013. ISBN 9781450324045. doi: 10.1145/2502081.2502184.

Jules Françoise, Norbert Schnell, Riccardo Borghesi, Frédéric Bevilacqua, and Place Igor Stravinsky. Probabilistic Models for Designing Motion and Sound Relationships. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 287–292, 2014.

Jules Françoise, Frédédric Bevilacqua, and Thecla Schiphorst. Supporting User Interaction with Machine Learning through Interactive Visualizations. In Marco Gillies and Rebecca Fiebrink, editors, *ACM SIGCHI Workshop on Human-Centered Machine Learning*, San Jose, 2016. ACM. URL http://hcml2016.goldsmithsdigital.com/program/.

Nikolaus Franke and Eric Von Hippel. Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software. *Research Policy*, 32(7):1199–1215, 2003. ISSN 00487333. doi: 10.1016/S0048-7333(03)00049-0.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley, 1994. ISBN ISBN 0-201-63361-2.

William Gaver. What should we expect from research through design? In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 937–946, 2012. ISBN 145031015X.

Nicholas Gillian and Joseph A Paradiso. The Gesture Recognition Toolkit. *Journal of Machine Learning Research*, 15:3483–3487, 2014. ISSN 15337928. doi: 10.13140/2.1.4216.2886. URL `http://jmlr.org/papers/v15/gillian14a.html`.

Thomas Green. Cognitive dimensions of notations. In *Proceedings of the fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and computers V*, pages 443–460, 1989. ISBN 0-521-38430-3. doi: citeulike-article-id:634333. URL `http://www.cl.cam.ac.uk/users/afb21/CognitiveDimensions/papers/Green1989.pdf`.

T.R.G. Green and M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages & Computing*, 7(2):131–174, 1996. ISSN 1045926X. doi: 10.1006/jvlc.1996.0009. URL `http://linkinghub.elsevier.com/retrieve/pii/S1045926X96900099`.

Saul Greenberg and Bill Buxton. Usability evaluation considered harmful (some of the time). *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 111, 2008. ISSN 978-1-60558-011-1. doi: 10.1145/1357054.1357074. URL `http://portal.acm.org/citation.cfm?doid=1357054.1357074`.

Mick Grierson. Maximillian: A cross platform C++ Audio Synthesis Library for artists learning to program, 2010.

José Guerreiro, Raúl Martins, Hugo Silva, André Lourenço, and Ana Fred. BITalino: A Multimodal Platform for Physiological Computing. *Proc. of the 10th ICINCO Conf*, pages 500–506, 2013. doi: 10.5220/0004594105000506.

Mark A Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18, 2009. ISSN 19310145. doi: 10.1145/1656274.1656278. URL `http://portal.acm.org/citation.cfm?id=1656274.1656278`.

Kim Halskov and Peter Dalsgård. Inspiration card workshops. *Proceedings of the 6th ACM conference on Designing Interactive systems - DIS '06*, page 2, 2006. ISSN 1470-3572. doi: 10.1145/1142405.1142409. URL `http://portal.acm.org/citation.cfm?doid=1142405.1142409`.

Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, page 145, 2007. ISBN 9781595935939. doi: 10.1145/1240624.1240646. URL `http://portal.acm.org/citation.cfm?doid=1240624.1240646`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning - Data Mining, Inference and Prediction.* 2nd ed edition, 2017. ISBN 978-0-387-84857-0. doi: 10.1007/b94608.

Gillian R Hayes. The Relationship of Action Research to Human-computer Interaction. *ACM Trans. Comput.-Hum. Interact.*, 18(3):15:1—-15:20, aug 2011. ISSN 1073-0516. doi: 10.1145/1993060.1993065. URL http://doi.acm.org/10.1145/1993060.1993065.

Michi Henning. API design matters. *Communications of the ACM*, 52(5):46, 2009. ISSN 00010782. doi: 10.1145/1506409.1506424. URL http://portal.acm.org/citation.cfm?doid=1506409.1506424.

Kathryn Herr. *The Action Research Dissertation: A Guide for Students and Faculty.* Sage Publications, Inc., California, 2005. ISBN 0-7619-2990-8.

L E Holmquist. Bootlegging: Multidisciplinary brainstorming with cut-ups. In *Tenth Conference on Participatory Design, PDC 2008*, pages 14–17, 2008. ISBN 0981856101.

Lars Erik Holmquist. Grounded Innovation: Strategies for Creating Digital Products by Lars Erik Holmquist, 2013. ISSN 0163-5948. URL http://doi.acm.org/10.1145/2413038.2413054.

K. Holtzblatt, J. B. Wendell, and S. Wood. *Rapid Contextual Design: A How-To Guide to Key Techniques for User-Centered Design.* Morgan Kaufmann Publishers Inc., San Francisco, CA, 2004.

Eric Horvitz. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems.*, pages 159–166. ACM, 1999. ISBN 0201485591. doi: 10.1145/302979.303030.

Alejandro Jaimes and Nicu Sebe. Multimodal human–computer interaction: A survey. *Computer Vision and Image Understanding*, 108(1-2):116–134, oct 2007. ISSN 10773142. doi: 10.1016/j.cviu.2006.10.019. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314206002335.

Robert J. Jasper and Leslie M. Blaha. Interface metaphors for interactive machine learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10284 LNAI:521–534, 2017. ISSN 16113349. doi: 10.1007/978-3-319-58628-1_39.

Lars Bo Jeppesen and Lars Frederiksen. Why Do Users Contribute to Firm-Hosted User Communities? The Case of Computer-Controlled Music Instruments. *Organization Science*, 17(July 2015):45–63, 2006. ISSN 1047-7039. doi: 10.1287/orsc.1050.0156.

Sinno Jialin and Qiang Yang. A Survey on Transfer Learning. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 1(10):1–15, 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191.

CW Kirkwood. Decision tree primer. *Department of Supply Chain Management, Arizona ...*, pages 1–18, 2002. ISSN 1460-2350. doi: 10.1093/humrep/deq210. URL http://203.64.187.41/htdocs-41/download-area/DATA-MINING/Apriori/D-T/DecisionTreePrimer-Front.pdf.

Ned Kock. The Three Threats of Organizational Action Research: Their nature and related antidotes. In Ned Kock, editor, *Information Systems Action Research: An Applied View of Emerging Concepts and Methods*, pages 97–129. Springer Science+Business Media, 2007. ISBN 9780387360591. URL `http://www.springer.com/business+{&}+management/business+information+systems/book/978-0-387-36059-1`.

Ned Kock. Action Research: Its Nature and Relationship to Human-Computer Interaction, 2014. URL `https://www.interaction-design.org/encyclopedia/action{_}research.html`.

Ron Kohavi and Foster Provost. Glossary of Terms. *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process - Machine Learning*, 30:271–274, 1998.

Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, 17:1–5, 2016. ISSN 15337928. doi: 10.1016/0022-1694(93)90238-5.

Max Kuhn, Jed Wing, Steve Weston, and Andre Williams. The Caret Package. *Gene Expression*, 2007. URL `http://topepo.github.io/caret/index.html`.

Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. Structured Labeling to Facilitate Concept Evolution in Machine Learning. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 3075–3084, 2014. doi: 10.1145/2556288.2557238. URL `http://dl.acm.org/citation.cfm?doid=2556288.2557238`.

Karim R. Lakhani and Eric Von Hippel. How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6):923–943, 2003. ISSN 00487333. doi: 10.1016/S0048-7333(02)00095-1.

Kurt Lewin. Action Research and Minority Problems. *Journal of Social Issues*, 2(4):34–46, 1946. ISSN 00224537. doi: 10.1111/j.1540-4560.1946.tb02295.x. URL `http://onlinelibrary.wiley.com/doi/10.1111/j.1540-4560.1946.tb02295.x/full{%}5Cnhttp://libezproxy.open.ac.uk/login?url=http://search.ebscohost.com/login.aspx?direct=true{&}db=cja{&}AN=16473285{&}site=eds-live{&}scope=site{%}5Cnhttp://doi.wiley.com/10.1111/j.1540-4560.194`.

Henry Lieberman, Fabio Paternó, Markus Klann, and Volker Wulf. End-user development: An emerging paradigm. *End user development*, 9:1–8, 2006. ISSN 09376429. doi: 10.1007/1-4020-5386-X. URL `http://link.springer.com/chapter/10.1007/1-4020-5386-X{_}1`.

Silvia Lindtner, Garnet D Hertz, and Paul Dourish. Emerging sites of HCI innovation: Hackerspaces, Hardware Startups & Incubators. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 439–448, 2014. ISBN 9781450324731. doi: 10.1145/2556288.2557132. URL `http://doi.acm.org/10.1145/2556288.2557132{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2556288.2557132`.

WE Mackay. Using Video to Support Interaction Design. *DVD Tutorial, CHI*, 2002. URL http://www.cs.ubc.ca/{~}cs544/video/Mackay-using-video-usletter.pdf.

Allan MacLean, Richard M. Young, Victoria M.E. Bellotti, and Thomas P. Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction*, 6 (3-4):201–250, 1991. ISSN 15327051. doi: 10.1080/07370024.1991.9667168.

Stephen MacNeil, Johanna Okerlund, and Celine Latulipe. Dimensional Reasoning and Research Design Spaces. *Proc. of ACM Creativity and Cognition*, pages 367–379, 2017. doi: 10.1145/3059454.3059472. URL http://dl.acm.org/citation.cfm?doid=3059454.3059472.

Oscar Mayor, Quim Llimona, Marco Marchini, Panos Papiotis, and Esteban Maestre. repoVizz: A Framework for Remote Storage, Browsing, Annotation, and Exchange of Multi-modal D ata. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 415–416, 2013. ISBN 9781450324045. doi: 10.1145/2502081.2502247. URL http://phenicx.upf.edu/node/107.

Carver Mead and Lynn Conway. *Introduction to VLSI systems.* Addison-Wesley, Reading, Mass., 1980. ISBN 0201043580.

Sebastián Mealla, Hugo Silva, Norbert Schnell, Francisco Bernardo, Carles Julià, Fabian Renn-Giles, Frédéric Belvilacqua, Michael Zbyszyński, Mick Grierson, Rebecca Fiebrink, Panagiotis Papiotis, Joseph Larralde, and Xavier Boissarie. D2.3 Design Guidelines for the RAPID-API. Technical report, RAPID-MIX, 2016.

David A Mellis, Ben Zhang, and Audrey Leung. Machine Learning for Makers : Interactive Sensor Data Classification Based on Augmented Code Examples. volume 2, pages 1213–1225, 2017. ISBN 9781450349222. doi: 10.1145/3064663.3064735.

T.M. Mitchell. *Machine Learning.* McGraw-Hill, New York, NY, USA, 1997.

Tom M Mitchell. The Discipline of Machine Learning. *Machine Learning*, 17(July):1–7, 2006. ISSN 0264-0414. doi: 10.1080/026404199365326. URL http://www-cgi.cs.cmu.edu/{~}tom/pubs/MachineLearningTR.pdf.

Andrew Monk. Lightweight Techniques to Encourage Innovative User Interface Design. In Larry E. Wood, editor, *User Interface Design: Bridging the Gap from User Requirements to Design.* CRC Press LLC, 2007.

David C Mowery and Bhaven N Sampat. Universities in National Innovation Systems. In Jan Fagerberg, David C Mowery, and Richard R. Nelson, editors, *The Oxford Handbook of Innovation*, pages 1–38. 1980.

Brad Myers, Scott E. Hudson, and Randy Pausch. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction*, 7(1):3–28, 2000. ISSN 10730516. doi: 10.1145/344949.344959. URL http://portal.acm.org/citation.cfm?doid=344949.344959.

Brad A. Myers and Jeffrey Stylos. Improving API usability. *Communications of the ACM*, 59(6): 62–69, 2016. ISSN 00010782. doi: 10.1145/2896587. URL `http://dl.acm.org/citation.cfm?doid=2942427.2896587`.

Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. What Makes a Good Code Example? A Study of Programming Q & A in StackOverflow. pages 25–34, 2012.

Nesta. The Innovation Index. *Technology*, 82(November):1–28, 2009. URL `http://www.nesta.org.uk/library/documents/innovation-index.pdf`.

Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1994.

Donald A. Norman. The way I see it: Technology first, needs last. *Interactions*, 17(2):38, 2010a. ISSN 10725520. doi: 10.1145/1699775.1699784. URL `http://doi.acm.org/10.1145/1699775.1699784{%}5Cnhttp://dl.acm.org/ft{_}gateway.cfm?id=1699784{&}type=pdf`.

Donald A. Norman. The Research-Practice Gap : The Need for Translational Developers. *Interactions*, (July-August):9–12, 2010b. ISSN 10725520. doi: 10.1145/1806491.1806494.

Donald A. Norman. *The Design of Everyday Things*. MIT Press, 2013.

Donald A. Norman and Stephen W. Draper. *User Centered System Design: New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986. ISBN 0898597811.

Juliet Norton, Chadwick A. Wingrave, and Joseph J. LaViola. Exploring strategies and guidelines for developing full body video game interfaces. *Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10*, pages 155–162, 2010. doi: 10.1145/1822348.1822369. URL `http://dl.acm.org/citation.cfm?id=1822348.1822369`.

OECD. OECD Innovation Strategy 2015 An Agenda for Policy Action. *OECD Reviews of Innovation Policy*, (June):395–423, 2015. doi: 10.1787/9789264039827-en. URL `http://www.oecd.org/mcm/documents/OECD-Innovation-Strategy-2015-CMIN2015-7.pdf{%}5Cnhttp://www.oecd-ilibrary.org/science-and-technology/oecd-reviews-of-innovation-policy-china-2008/framework-conditions-for-innovation{_}9789264039827-12-en`.

Dan Olsen. Evaluating user interface systems research. *20th annual ACM symposium on User interface software and technology (UIST'07)*, pages 251–258, 2007. doi: 10.1145/1294211.1294256.

Wanda J Orlikowski and C Suzanne Iacono. Research Commentary: Desperately Seeking the "IT" in IT Research—A Call to Theorizing the IT Artifact. *Information Systems Research*, 12(2): 121–134, 2001. ISSN 1047-7047. doi: 10.1287/isre.12.2.121.9700.

Adam Parkinson, David Cameron, and Atau Tanaka. HapticWave: presenting the multiple voices, artefacts and materials of a design research project. In *Proceedings of the 2nd Biennial Research Through Design Conference*, Cambridge, 2015. doi: 10.6084/m9.figshare.1327979.HapticWave.

Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. Investigating statistical machine learning as a tool for software development. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, (May):667, 2008. doi: 10.1145/1357054.1357160. URL http://portal.acm.org/citation.cfm?doid=1357054.1357160.

Kayur Patel, Naomi Bancroft, Steven M Drucker, James Fogarty, Andrew J Ko, and James Landay. Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 37–46, 2010. ISBN 1450302718. doi: 10.1145/1866029.1866038.

Marian Petre. Cognitive dimensions 'beyond the notation'. *Journal of Visual Languages and Computing*, 17(4):292–301, 2006. ISSN 1045926X. doi: 10.1016/j.jvlc.2006.04.003.

Michael Polanyi. *The Tacit Dimension.* University of Chicago Press, Chicago, 2009.

Jenny Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design: Beyond Human-Computer Interaction.* 4th edition, 2015.

R. N. Rapoport. Three Dilemmas in Action Research. *Human Relations*, 23(6):499–513, 1970. ISSN 1059-6011. doi: 0803973233. URL http://raj.sagepub.com/lookup/doi/10.1177/2153368714567577{%}5Cnhttp://dx.doi.org/10.1023/A:1007521427059{%}5Cnhttp://onlinelibrary.wiley.com.prox.lib.ncsu.edu/doi/10.1111/j.1728-4457.2005.00079.x/pdf{%}5Cnhttp://www.tandfonline.com/doi/abs/10.1080/00380237.2002.1057.

Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. Design Principles for Tools to Support Creative Thinking. Technical report, 2005.

F.E. Ritter, G.D. Baxter, and E.F. Churchill. *Foundations for Designing User-Centered Systems: What System Designers Need to Know about People.* Springer London Heidelberg New York Dordrecht, London, 2014.

Martin P. Robillard and Robert Deline. A field study of API learning obstacles. *Empirical Software Engineering*, 16(6):703–732, 2011. ISSN 13823256. doi: 10.1007/s10664-010-9150-8.

Colin Robson. *Real world research*, volume 3. Wiley Chichester, 2011.

Everett M. Rogers. *Diffusion of Innovations.* Simon and Schuster, 5th editio edition, 2003.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, New Jersey, 3rd ed. edition, 2009.

Advait Sarkar. Constructivist Design for Interactive Machine Learning. *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 1467–1475, 2016. doi: 10.1145/2851581.2892547. URL http://dx.doi.org/10.1145/2851581.2892547.

Robert E. Schapire. A Short Introduction to Boosting. In *Journal of Japanese Society for Artificial Intelligence*, volume 14, pages 771–780, 1999. ISBN 3540440119. doi: citeulike-article-id:765005. URL `http://arxiv.org/abs/1508.01136`.

Thomas Scheller and Eva Kühn. Automated measurement of API usability: The API Concepts Framework. *Information and Software Technology*, 61(February):145–162, 2015. ISSN 09505849. doi: 10.1016/j.infsof.2015.01.009. URL `http://dx.doi.org/10.1016/j.infsof.2015.01.009`.

Norbert Schnell. Project CoSiMa, 2014. URL `http://cosima.ircam.fr`.

Norbert Schnell, Axel Röbel, Diemo Schwarz, Geoffroy Peeters, and Riccardo Borghesi. MUBU and Friends – Assembling tools for content based real-time interactive audio processing in MAX/MSP. In *Proceedings of International Computer Music Conference*, volume 2009, pages 423–426, 2009. ISBN 9780971319271.

Maung K Sein, Ola Henfridsson, Matti Rossi, and Rikard Lindgren. Action Design Research. *MIS Quarterly*, 35(1):37–56, 2011. ISSN 02767783. doi: DOI:10.2307/23043488.

Burr Settles. Active Learning Literature Survey. Technical report, University of Wisconsin, Madison, 2009.

Herbert A Simon. The Structure of Ill Structured Problems. *Artificial Intelligence*, 4(3-4):181–201, 1973. URL `http://www.public.iastate.edu/{~}cschan/235/6{_}Simon{_}Ill{_}defined{_}problem.pdf`.

G. Ryan Spain. Key Research Findings. Technical Report November 2015, 2017. URL `https://dzone.com/storage/assets/6907133-dzone-guide-artificialintelligence-2017.pdf`.

Evan R Sparks, Shivaram Venkataraman, Tomer Kaftan, Michael Franklin, and Benjamin Recht. KeystoneML : Optimizing Pipelines for Large-Scale Advanced Analytics. 2016.

Gerald Susman and Roger D Evered. An Assessment of the Scientific Merits of Action Research. *Administrative Science Quarterly*, 23(4):582–603, 1978.

Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S. Tan. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1283–1292, 2009. ISSN 9781605582467. doi: 10.1145/1518701.1518895. URL `http://portal.acm.org/citation.cfm?id=1518895`.

Atau Tanaka and Adam Parkinson. Haptic Wave: A Cross-Modal Interface for Visually Impaired Audio Producers. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2150–2161, 2016. doi: 10.1145/2858036.2858304. URL `http://doi.acm.org/10.1145/2858036.2858304`.

Thalmic Labs Inc. Myo. URL `https://www.myo.com/`.

Eric L Trist. *Action Research and Adaptive Planning*, chapter Engaging w, pages 223–236. Springer US, Boston, MA, 1976. ISBN 978-1-4613-4262-5. doi: 10.1007/978-1-4613-4262-5_5. URL http://dx.doi.org/10.1007/978-1-4613-4262-5{_}5.

W. M. K. Trochim. Positivism & Post-Positivism, 2006. URL http://www.socialresearchmethods.net/kb/positvsm.php.

Jaroslav Tulach. *Practical API Design: Confessions of a JavaTM Framework Architect*. Apress, 2008. ISBN 978-1-4302-0973-7.

M. Turk. A random walk through Eigenspace. *IEICE Transactions on Information and Systems*, E84-D(12):1586–1595, 2001. ISSN 09168532.

Bill Venners and Bruce Eckel. The C# Design Process: A Conversation with Anders Hejlsberg, Part I, 2003. URL https://www.artima.com/intv/csdes2.html.

Eric von Hippel. Lead Users: An Important Source of Novel Product Concepts. *Management Science*, 32(7):791–805, 1986. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

Eric von Hippel. "Sticky information" and the locus of problem solving: implications for innovation. *Management Science*, 40(4):429–439, 1994. ISSN 0025-1909. doi: 10.1287/mnsc.40.4.429.

Eric von Hippel. *Democratizing innovation.* MIT Press, Cambridge, Massachussets, 2005. ISBN 0262002744. doi: 10.1111/j.1540-5885.2006.00192_2.x. URL http://discovery.ucl.ac.uk/630/.

Eric von Hippel, Stephen Flowers, Joeren De Jong, and Tanja Sinozic. Measuring user innovation in the UK: The importance of product creation by users. Technical Report April, 2010.

Eric A. von Hippel. User toolkits for innovation. *Journal of Product Innovation Management*, 35(forthcoming):63–64, 2001. ISSN 07376782. URL http://www.wiley.com/bw/journal.asp?ref=0737-6782{&}site=1.

Karel Vredenburg, Ji-Ye Mao, Paul W. Smith, and Tom Carey. A survey of user-centered design practice. *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, (1):471, 2002. ISSN 10629432. doi: 10.1145/503457.503460. URL http://portal.acm.org/citation.cfm?doid=503376.503460.

Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H Witten. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3): 281–292, 2001. ISSN 10715819. doi: 10.1006/ijhc.2001.0499. URL http://www.sciencedirect.com/science/article/pii/S1071581901904999.

Robert Watson. Applying the Cognitive Dimensions of API Usability to Improve API Documentation Planning. *Proceedings of the 32nd ACM International Conference on The Design of Communication CD-ROM*, pages 2–3, 2014. doi: 10.1145/2666216.2666239.

Karl E Wiegers. Humanizing Peer Reviews. Technical Report April, 2002. URL `http://processimpact.com/articles/humanizing{_}reviews.pdf`.

Chamila Wijayarathna, Nalin Arachchilage, and Jill Slay. A Generic Cognitive Dimensions Questionnaire to Evaluate the Usability of Security APIs. In T. Tryfonas, editor, *Human Aspects of Information Security, Privacy and Trust. HAS 2017. Lecture Notes in Computer Science*, volume 10292, pages 160–173. Springer, Cham, 2017. ISBN 978-3-319-58459-1. doi: DOI:10.1007/978-3-319-58460-711. URL `http://link.springer.com/10.1007/978-3-319-58460-7`.

Jacob O Wobbrock. Seven Research Contributions in HCI. *Studies*, pages 1–6, 2012. URL `http://faculty.washington.edu/wobbrock/pubs/Wobbrock-2012.pdf`.

Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya Parameswaran. Helix: Accelerating Human-in-the-loop. *Machine Learning. PVLDB*, 11(12):1958–1961, 2018a. doi: 10.14778/3229863.3236234.

Doris Xin, Litian Ma, Shuchen Song, and Aditya Parameswaran. How Developers Iterate on Machine Learning Workflows – A Survey of the Applied Machine Learning Literature. 2018b. doi: arXiv:1803.10311v2. URL `http://arxiv.org/abs/1803.10311`.

Qian Yang, Jina Suh, Nan Chen Chen, and Gonzalo Ramos. Grounding interactive machine learning tool design in how non-experts actually build models. *DIS 2018 - Proceedings of the 2018 Designing Interactive Systems Conference*, pages 573–584, 2018. doi: 10.1145/3196709.3196729.

Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Hu Yi-Qi, Li Yu-Feng, Tu Wei-Wei, Yang Qiang, and Yu Yang. Taking Human out of Learning Applications: A Survey on Automated Machine Learning. *arXiv preprint arXiv:1810.13306*, pages 1–15, 2018. URL `www.aaai.orghttp://arxiv.org/abs/1810.13306`.

Alon Zakai. Emscripten: An LLVM-to-JavaScript Compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. ACM, 2011.

Michael Zbyszyński, Mick Grierson, Matthew Yee-king, and Leon Fedden. Write once run anywhere revisited: machine learning and audio tools in the browser with C++ and emscripten. In *Web Audio Conference 2017*, pages 1–5. Centre for Digital Music, Queen Mary University of London, 2017.

Ce Zhang, Arun Kumar, and Christopher Ré. Materialization Optimizations for Feature Selection Workloads. *ACM Transactions on Database Systems*, 41(1):1–32, 2016. ISSN 03625915. doi: 10.1145/2877204. URL `http://dl.acm.org/citation.cfm?doid=2897141.2877204`.

Martin Zinkevich. Rules of Machine Learning: Best Practices for ML Engineering, 2018.

Appendix A

Principles of Canonical Action Research (Davidson et al., 2004)

A.1 Principle of the Researcher–Client Agreement

a) Did both the researcher and the client agree that CAR was the appropriate approach for the organizational situation?

b) Was the focus of the research project specified clearly and explicitly?

c) Did the client make an explicit commitment to the project?

d) Were the roles and responsibilities of the researcher and client organisation members specified explicitly?

e) Were project objectives and evaluation measures specified explicitly?

f) Were the data collection and analysis methods specified explicitly?

A.2 Principle of the Cyclical Process Model

a) Did the project follow the CPM or justify any deviation from it?

b) Did the researcher conduct an independent diagnosis of the organizational situation?

c) Were the planned actions based explicitly on the results of the diagnosis?

d) Were the planned actions implemented and evaluated?

e) Did the researcher reflect on the outcomes of the intervention?

f) Was this reflection followed by an explicit decision on whether or not to proceed through an additional process cycle?

g) Were both the exit of the researcher and the conclusion of the project due to either the project objectives being met or some other clearly articulated justification?

A.3 Principle of Theory

a) Were the project activities guided by a theory or set of theories?

b) Was the domain of investigation, and the specific problem setting, relevant and significant to the interests of the researcher's community of peers as well as the client?

c) Was a theoretically based model used to derive the causes of the observed problem?

d) Did the planned intervention follow from this theoretically based model?

e) Was the guiding theory, or any other theory, used to evaluate the outcomes of the intervention?

A.4 Principle of Change through Action

a) Were both the researcher and client motivated to improve the situation?

b) Were the problem and its hypothesized cause(s) specified as a result of the diagnosis?

c) Were the planned actions designed to address the hypothesized cause(s)?

d) Did the client approve the planned actions before they were implemented?

e) Was the organization situation assessed comprehensively both before and after the intervention?

f) Were the timing and nature of the actions taken clearly and completely documented?

A.5 Principle of Learning through Reflection

a) Did the researcher provide progress reports to the client and organizational members?

b) Did both the researcher and the client reflect upon the outcomes of the project?

c) Were the research activities and outcomes reported clearly and completely?

d) Were the results considered in terms of implications for further action in this situation?

e) Were the results considered in terms of implications for action to be taken in related research domains?

f) Were the results considered in terms of implications for the research community (general knowledge, informing/re-informing theory)?

g) Were the results considered in terms of the general applicability of CAR?

Appendix B

Summary of available techniques for deployment in UCD actions

Technique	Purpose of study	Target	Data output	Cost	Project stage
Ethnographic interviews, and observation	Assessment of user needs and insights; Identify design opportunities for SMEs; Interpret user behaviour	Small groups; SME stakeholders; End users	Quantitative and qualitative	High	Early stage and throughout the project
Online surveys, offline surveys	Assessment of user needs, insights and feedback	Large groups; End users	Quantitative and qualitative	Medium	Early stage and throughout the project
Brainstorming Video-brainstorming, Bootlegging	Assessment of user needs and insights; Ideation, identification of scenarios and design themes	Small groups; SME stakeholders	Qualitative	Low	Early stage
Sketching Storyboarding	Assessment of user needs and insights; Ideation, identification of scenarios and design themes	Medium size groups; SME stakeholders; End users	Qualitative	Low	Early stage
Design workbooks	Collect and document ideas and perspectives	Small groups; SME stakeholders	Qualitative	Low	Beginning of project

Technique	Purpose of study	Target	Data output	Cost	Project stage
Low-fidelity prototyping	Assess prototype compatibility with SMEs production roadmap	Small groups; SME stakeholders	Qualitative	Low	At initial stage and throughout project
High-fidelity prototyping	Testing prototypes with SMEs and assess their compatibility with SMEs production roadmap	Small groups; SME stakeholders; End users;	Qualitative	High	Mid point in the project
Usability tests	Identify usability problems; Benchmarking and validation of MIX interfaces	5 to 12 participants; End users	Quantitative and qualitative	High	Mid-to-final stage
Hacking, Do-It-Yourself, Appropriation	Identify new affordances in existing technologies and unexpected uses	Medium size groups; End users groups; SME stakeholders	Qualitative	Low	Final point in the project
Logging	Identify patterns of use, usability and technology problems both for API and MIX products	Small groups; SME stakeholders	Quantitative and qualitative	Medium	Mid point and final stages
Focus groups	Evaluation of prototypes with end users; extract quantifiable UX metrics	Small groups; SME stakeholders; End users	Qualitative	Low	Final stage
Cultural probes, Technology probes	Explore users' culture, attitudes towards environment and use of new media	Small groups; SME stakeholders; End users	Qualitative	Medium	Early stage
Longitudinal studies	Studying technology engagement over longer periods of time	Variable size groups; SME stakeholders; End users	Quantitative and qualitative	High	Throughout the entire project

Appendix C

Co-design workshop with RAPID-MIX stakeholders

rapidmix

CO-DESIGN WORKSHOP

Tuesday, June 16th 2015

Universitat Pompeu Fabra
Roc Boronat 138,
Room 52.325

C.1 Co-design themes identified in Paris meeting May 20-22, 2015, at IRCAM

1. ***Biosignal repository*** *applied to expressive performance (BITalino, RepoVizz, Wekinator)*
2. ***Processing in the hardware.*** *What would change? which are the specificities? which are the possible applications and constraints? which would be specifications*

for the API in case of embedded software ()

3. ***Designing meaningful expressive interactions*** *(i.e. for reactable running)*
4. ***Exploration of big sound databases*** *to find meaningful content (i.e. freesound)*

5. *What kind of* ***web applications*** *can we prototype within the project?*
6. ***Acquisition and analysis of multiple behavioural inputs****, coming from the environment or the user (using BITalino?) (Orbe)*
7. ***Providing interoperability*** *between Orbe's backend and repovizz's frontend*
8. ***Tagging sounds with geolocation*** *(using Freesound API) OR taking advantage of already geotagged sounds on Freesound*

(Excerpt from meeting minutes of RAPID-MIX consortium meeting at IRCAM, Paris, May 20-22, 2015)

C.2 Co-design workshop brief

C.2.1 Welcome

Welcome to the Rapid Mix Co-Design Day! This is the first in a series of technology brainstorming sessions what will help us generate ideas for innovative multimodal expressive products – the MIX in Rapid Mix, and to help us design a robust application programming interface (the API in Rapid) to support these product ideas. In this first workshop, we welcome your participation as members of the Rapid Mix EU project consortium. Rapid Mix is a 3-year project funded by an EC Horizon2020 "Innovation Action" to transfer knowledge and technology from research labs to industry. As the first participatory design activity in the first 6 months of the project, this will help us to create ideas for possible technology prototypes to be developed over the next six months.

We have a range of exciting technologies from Rapid Mix partners that we think can be combined in unique ways to enable exciting product ideas that would not have otherwise been possible. First, the Rapid Mix technologies themselves are cutting edge. Second, the brainstorming techniques we'll use today will unleash ideas that might not emerge from traditional technology or product development methods.

We have a fast paced programme of idea generation (ideation) and structured brainstorming that will get us thinking out of the box, identifying new and unusual applications for our technologies, and new combinations of the different technologies. The goal is to generate as many ideas as possible—there is no right or wrong, and there are no bad ideas! After generating as many ideas as possible, we'll filter and select, to focus *on a smaller number to elaborate further in usage scenarios and storyboards. We will enact these scenarios, even in absence of the technology itself, to understand how users may use the imaginary products, or how they might fit into broader contexts.*

The morning session will be open ended, and use the re-mix and mash-up metaphors from a technique called Bootlegging. The goal of this session is to generate ideas for products that use forms of interaction supported by Rapid Mix technologies that we would have never thought of otherwise. The afternoon session focuses on specific Rapid Mix technologies—code libraries, applications, hardware—to address a number of technology design challenges. We'll ask ourselves not just "what" we're building, but "why". Scenarios we build will not just be of end-user usage of end-products, but scenarios that depict the product design and development process, and ways in which Rapid Mix technologies, and the Rapid[-MIX] API could support the development of innovative interactive technology products. This will be fast paced, dynamic, and fun!

Francisco Bernardo, Rebecca Fiebrink, Sebastian Mealla & Atau Tanaka

C.2.2 Program

Morning Session: Generating Design Ideas

10:00 Intro

10:30 Generate Post-its (User)

10:40 Generate Post-its (Context)

10:50 Introduce Technology cards

11:00 Breakout groups

11:10 Bootlegging shuffle up and combination

11:20 Breakout group picks 4 use cases

11:30 Brainstorming the 4 use cases

12:10 Pick 1 use case and describe scenario through storyboarding

12:30 Presentation to whole group

13:00 Lunch break

Afternoon Session: Scenario building on core RM use cases as "Brainstorm Challenges"

14:00 Presentation of 8 challenges (2 min each)

14:20 Sign up

14:30 Breakout groups

14:45 Storyboarding exercise

15:30 Challenge 2 (5, 6, 7, 8)

15:45 Storyboarding exercise 2

16:30 Prepare presentations

17:00 Whole group

18:00-20:00 Freeform hands-on between developers

C.2.3 Breakout groups

Moderator: Atau Tanaka (GS)

Documentation: Francisco Bernardo (GS)

Group 1

Fred Bevilacqua (Coordinator, IRCAM)

Panos Papiotis (MTG)

Felix (ROLI)

Xavier (ORBE)

Group 2

Norbert Schnell (Coord IRCAM)

Carles F. Julia (MTG)

Jack Armitage (ROLI)

Tomek Jarolim (ORBE)

Group 3

Rebecca Fiebrink (Coordinator)

Sergi Jorda (MTG)

Emmanuel (IRCAM)

Fabian (ROLI)

Group 4

Sebastian Mealla (Coordinator)

Mick Grierson (GS)

Andres (RS)

Hugo Silva (PLUX)

C.3 Ideation and structured brainstorming cards

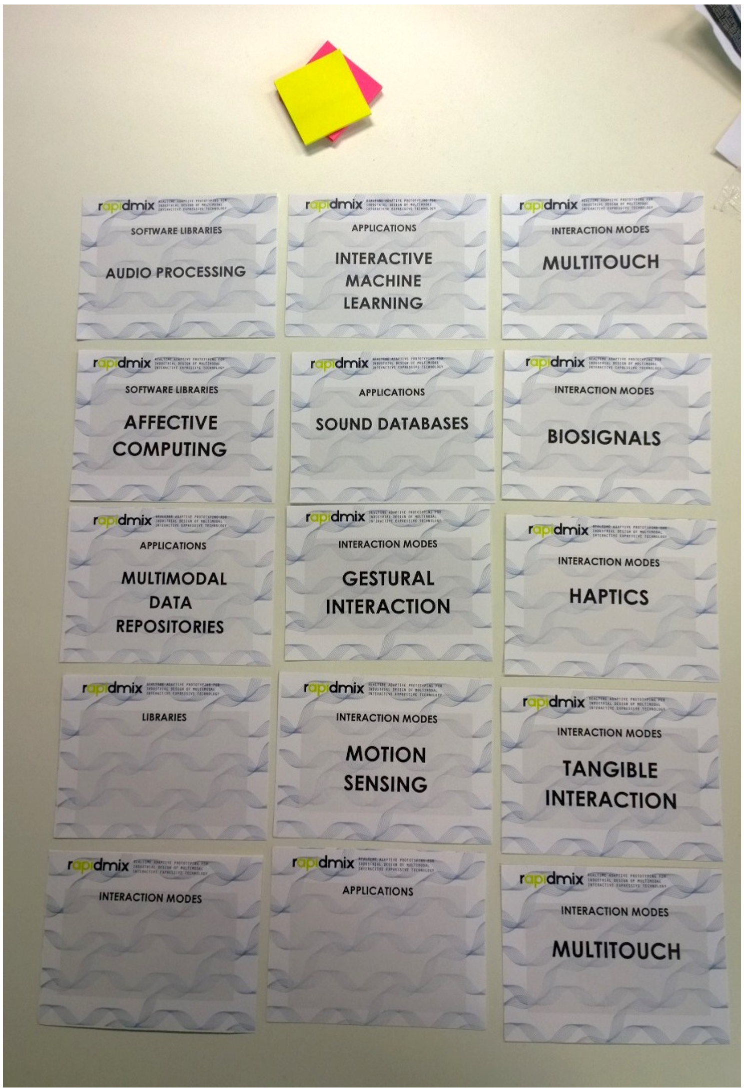


Figure C.1: Ideation cards

C.4 Storyboards from divergent design session

C.5 Affinity diagrams of potential users and contexts of use

C.6 Technology section in RAPID-MIX institutional website

C.7 Scenarios from convergent design session

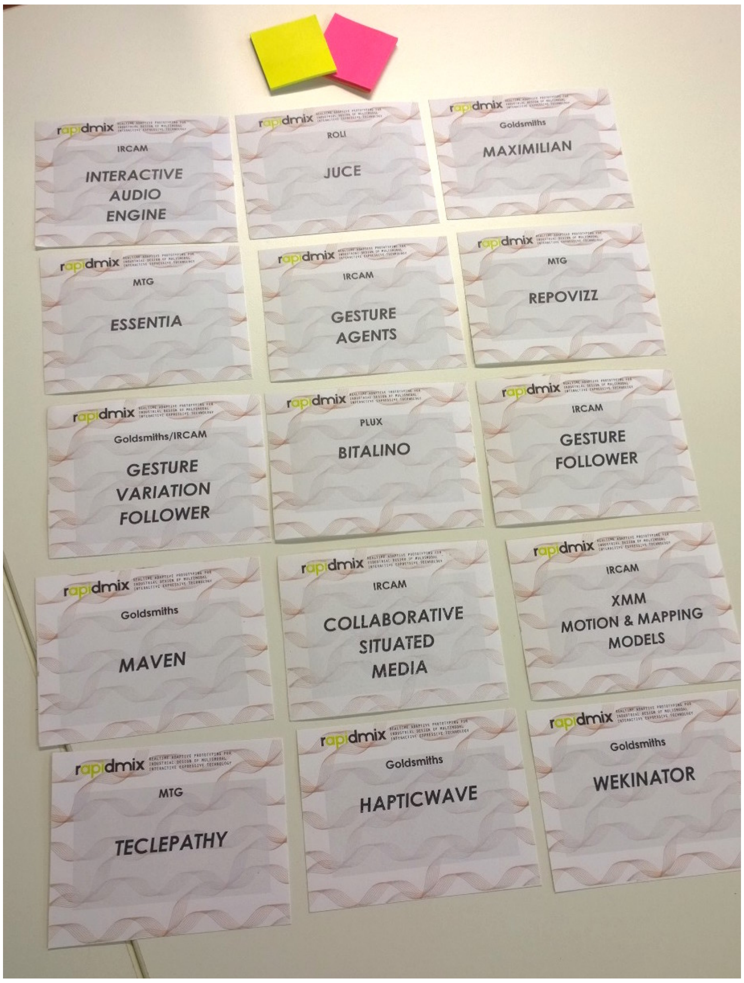


Figure C.2: Structured brainstorming cards

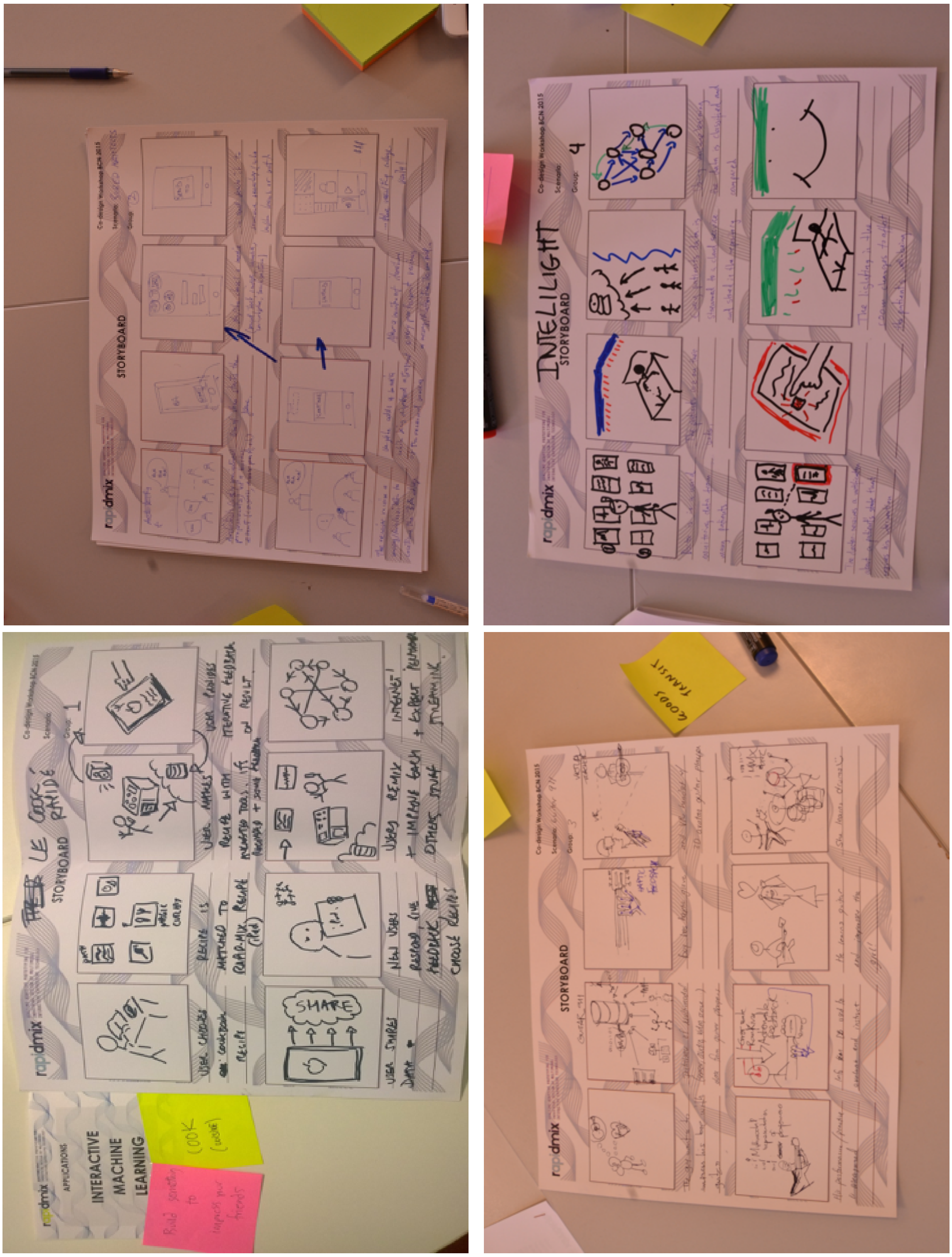


Figure C.3: Storyboards developed by the co-design groups (clockwise): a) *Le Cook Rapide*, b) *Bored Architects*, c) *Intellilight*, and d) *Guitar 911*

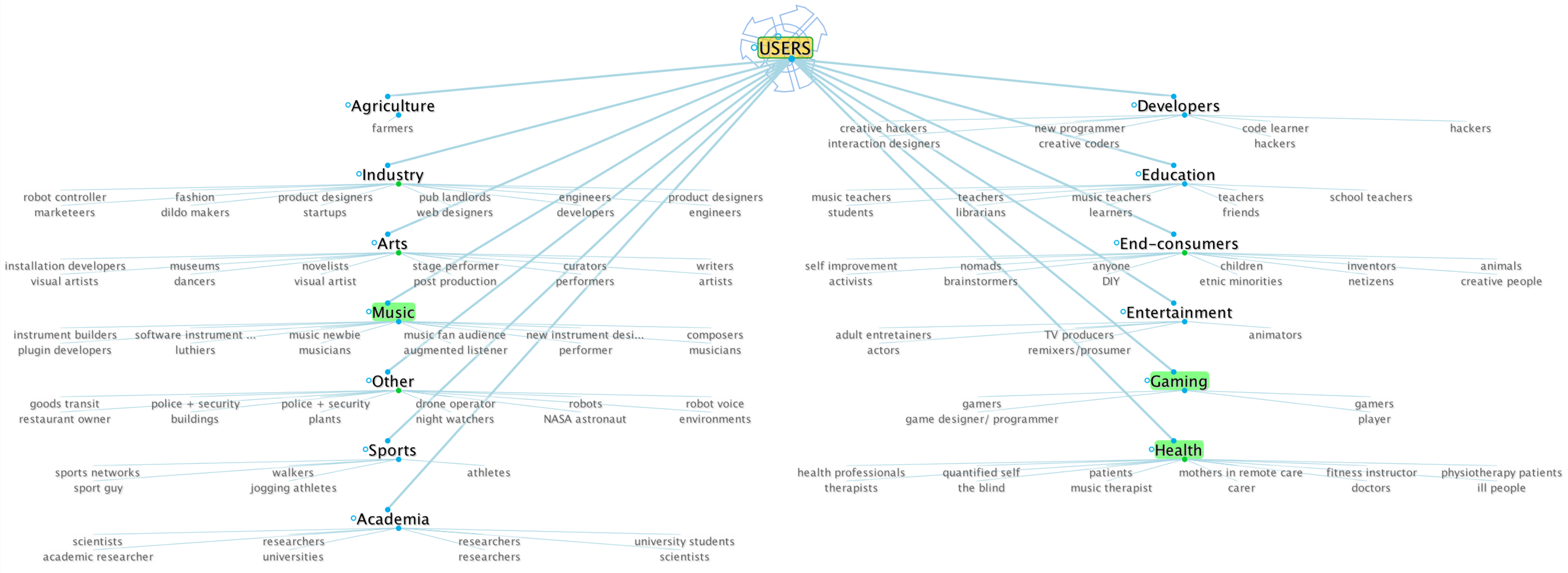


Figure C.4: Affinity diagram of potential RAPID-MIX users

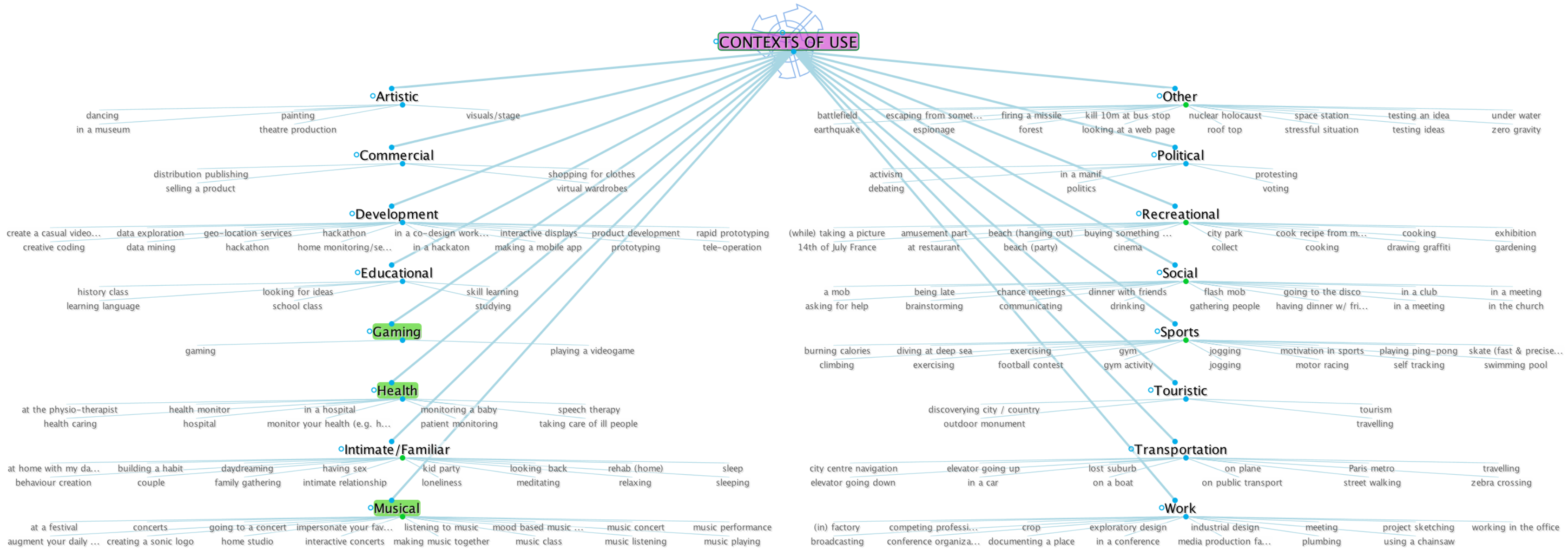
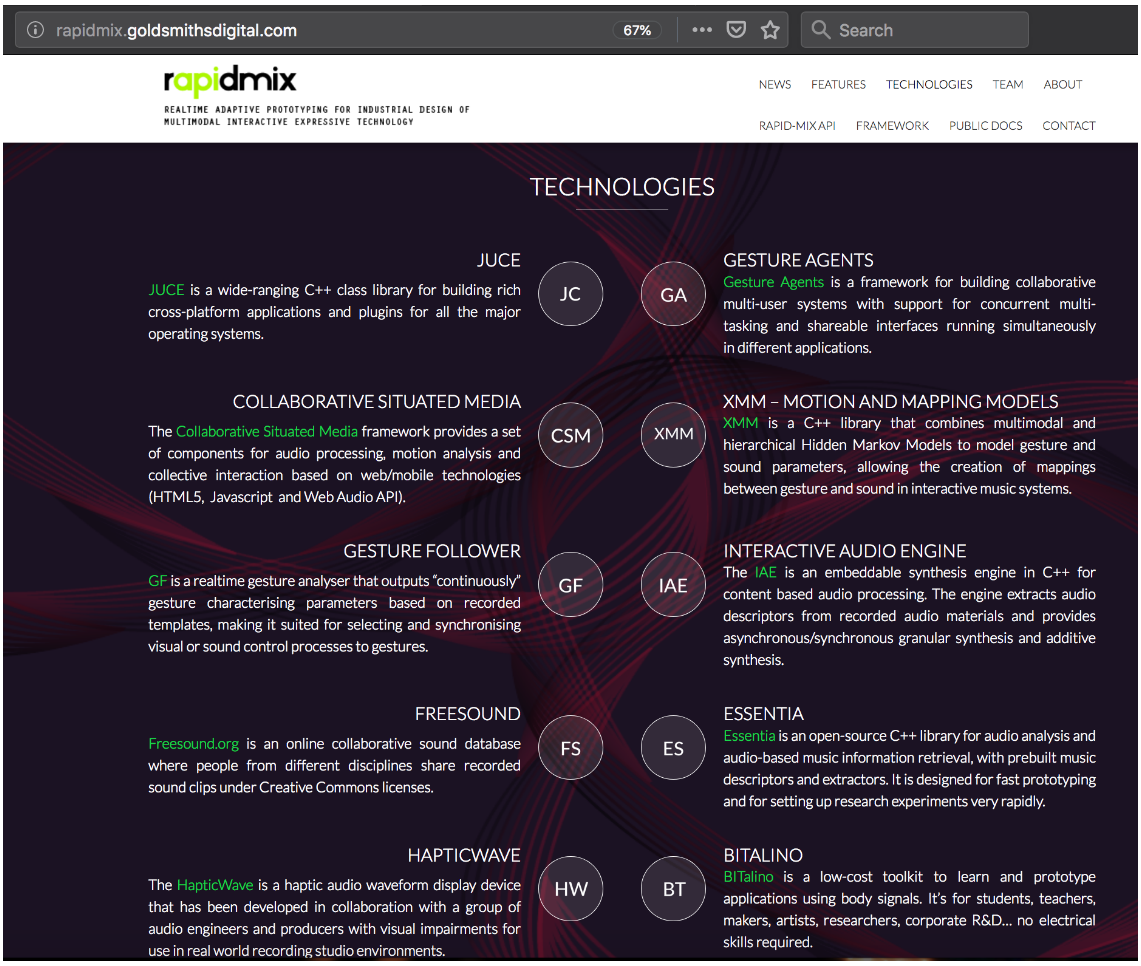


Figure C.5: Affinity diagram of potential contexts of use for RAPID-MIX technologies


rapidmix.goldsmithsdigital.com
67%
Search
rapidmix
REALTIME ADAPTIVE PROTOTYPING FOR INDUSTRIAL DESIGN OF MULTIMODAL INTERACTIVE EXPRESSIVE TECHNOLOGY
NEWS
FEATURES
TECHNOLOGIES
TEAM
ABOUT
RAPID-MIX API
FRAMEWORK
PUBLIC DOCS
CONTACT
JC
GA
CSM
XMM
GF
IAE
FS
ES
HW
BT

TECHNOLOGIES

JUCE

JUCE is a wide-ranging C++ class library for building rich cross-platform applications and plugins for all the major operating systems.

COLLABORATIVE SITUATED MEDIA

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

GESTURE FOLLOWER

GF is a realtime gesture analyser that outputs “continuously” gesture characterising parameters based on recorded templates, making it suited for selecting and synchronising visual or sound control processes to gestures.

FREESOUND

Freesound.org is an online collaborative sound database where people from different disciplines share recorded sound clips under Creative Commons licenses.

HAPTICWAVE

The HapticWave is a haptic audio waveform display device that has been developed in collaboration with a group of audio engineers and producers with visual impairments for use in real world recording studio environments.

GESTURE AGENTS

Gesture Agents is a framework for building collaborative multi-user systems with support for concurrent multi-tasking and shareable interfaces running simultaneously in different applications.

XMM – MOTION AND MAPPING MODELS

XMM is a C++ library that combines multimodal and hierarchical Hidden Markov Models to model gesture and sound parameters, allowing the creation of mappings between gesture and sound in interactive music systems.

INTERACTIVE AUDIO ENGINE

The IAE is an embeddable synthesis engine in C++ for content based audio processing. The engine extracts audio descriptors from recorded audio materials and provides asynchronous/synchronous granular synthesis and additive synthesis.

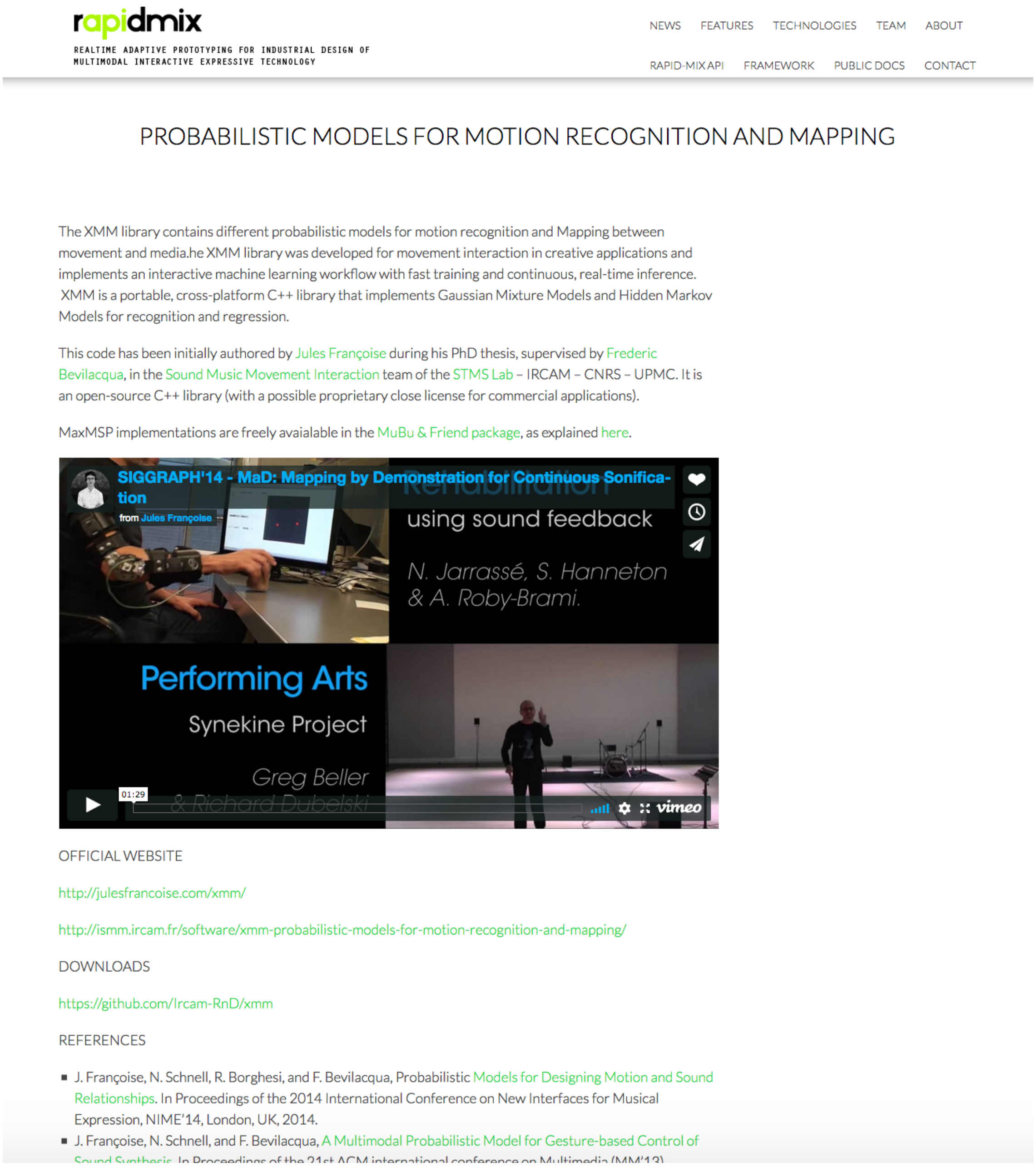
ESSENTIA

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval, with prebuilt music descriptors and extractors. It is designed for fast prototyping and for setting up research experiments very rapidly.

BITALINO

BITalino is a low-cost toolkit to learn and prototype applications using body signals. It's for students, teachers, makers, artists, researchers, corporate R&D... no electrical skills required.

Figure C.6: Technologies section of the RAPID-MIX institutional website


rapidmix
REALTIME ADAPTIVE PROTOTYPING FOR INDUSTRIAL DESIGN OF MULTIMODAL INTERACTIVE EXPRESSIVE TECHNOLOGY
NEWS
FEATURES
TECHNOLOGIES
TEAM
ABOUT
RAPID-MIX API
FRAMEWORK
PUBLIC DOCS
CONTACT

PROBABILISTIC MODELS FOR MOTION RECOGNITION AND MAPPING

The XMM library contains different probabilistic models for motion recognition and Mapping between movement and media.he XMM library was developed for movement interaction in creative applications and implements an interactive machine learning workflow with fast training and continuous, real-time inference. XMM is a portable, cross-platform C++ library that implements Gaussian Mixture Models and Hidden Markov Models for recognition and regression.

This code has been initially authored by Jules Françoise during his PhD thesis, supervised by Frederic Bevilacqua, in the Sound Music Movement Interaction team of the STMS Lab – IRCAM – CNRS – UPMC. It is an open-source C++ library (with a possible proprietary close license for commercial applications).

MaxMSP implementations are freely avaialable in the MuBu & Friend package, as explained here.

OFFICIAL WEBSITE

http://julesfrancoise.com/xmm/

http://ismm.ircam.fr/software/xmm-probabilistic-models-for-motion-recognition-and-mapping/

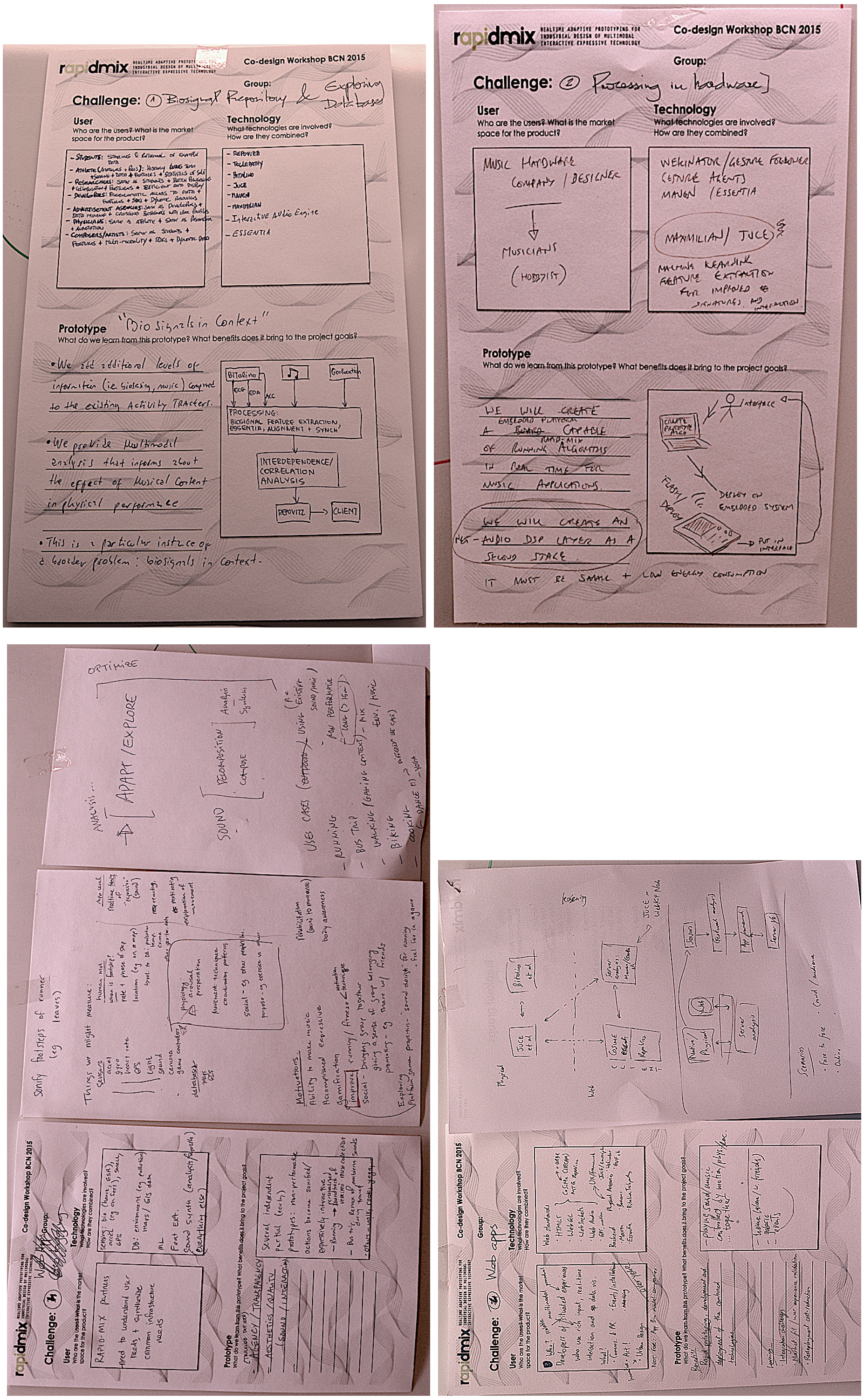
DOWNLOADS

https://github.com/Ircam-RnD/xmm

REFERENCES

- J. Françoise, N. Schnell, R. Borghesi, and F. Bevilacqua, Probabilistic Models for Designing Motion and Sound Relationships. In Proceedings of the 2014 International Conference on New Interfaces for Musical Expression, NIME'14, London, UK, 2014.
- J. Françoise, N. Schnell, and F. Bevilacqua, A Multimodal Probabilistic Model for Gesture-based Control of Sound Synthesis. In Proceedings of the 21st ACM international conference on Multimedia (MM'13).

Figure C.7: RAPID-MIX institutional website page with high-level summary about IRCAM's XMM



rapidmix REALTIME ADAPTIVE PROTOTYPING FOR INDUSTRIAL DESIGN OF MULTIMODAL INTERACTIVE EXPRESSIVE TECHNOLOGY

Co-design Workshop BCN 2015

Group:

Challenge: ① Biosignal Repository & Exploring Database

User
Who are the users? What is the market space for the product?

- STUDENTS: STORING & RETRIEVAL OF EXAMPLE DATA
- ATHLETE (AMATEURS + PROS): HISTORY DURING TIME + SHARING + DATA + FEATURES + STATISTICS OF SELF
- RESEARCHERS: SAME AS STUDENTS + BATCH PROCESSING + COLLABORATION + FEATURES + EFFICIENT DATA DISPLAY
- DEVELOPERS: PROGRAMMATIC ACCESS TO DATA + FEATURES + SDKS + DYNAMIC RESOURCES
- ADVERTISEMENT AGENCIES: SAME AS DEVELOPERS + DATA MINING + CROSSING BIOSIGNALS WITH USER PROFILES
- PHYSICIANS: SAME AS ATHLETE + SAME AS RESEARCHER + ANNOTATION
- COMPOSERS/ARTISTS: SAME AS STUDENTS + FEATURES + MULTI-MODALITY + SDKS + DYNAMIC DATA

Technology
What technologies are involved? How are they combined?

- REPOVIZZ
- TELEPATHY
- BITALINO
- JUCE
- MAVEN
- MAXIMILIAN
- Interactive Audio Engine
- ESSENTIA

Prototype "Bio Signals in Context"
What do we learn from this prototype? What benefits does it bring to the project goals?

- We add additional levels of information (ie. biosensing, music) compared to the existing Activity Trackers.
- We provide Multimodal analysis that informs about the effect of Musical Content in physical performance
- This is a particular instance of a broader problem: biosignals in context.

BITalino
ECG
EDA
ACC
♫
Geolocation
PROCESSING: BIOSIGNAL FEATURE EXTRACTION, ESSENTIA, ALIGNMENT + SYNCH
INTERDEPENDENCE/ CORRELATION ANALYSIS
REPOVIZZ
CLIENT

rapidmix REALTIME ADAPTIVE PROTOTYPING FOR INDUSTRIAL DESIGN OF MULTIMODAL INTERACTIVE EXPRESSIVE TECHNOLOGY

Co-design Workshop BCN 2015

Group:

Challenge: ② Processing in hardware]

User
Who are the users? What is the market space for the product?

MUSIC HARDWARE COMPANY / DESIGNER

MUSICIANS (HOBBYIST)

Technology
What technologies are involved? How are they combined?

WEKINATOR / GESTURE FOLLOWER
GESTURE AGENTS
MAVEN / ESSENTIA
MAXIMILIAN / JUCE
MACHINE LEARNING
FEATURE EXTRACTION FOR IMPROVED SIGNATURES AND INTERACTION.

Prototype
What do we learn from this prototype? What benefits does it bring to the project goals?

WE WILL CREATE A ~~BOARD~~ EMBEDDED PLATFORM CAPABLE OF RUNNING RAPID-MIX ALGORITHMS IN REAL TIME FOR MUSIC APPLICATIONS.

WE WILL CREATE AN NET-AUDIO DSP LAYER AS A SECOND STAGE.

IT MUST BE SMALL + LOW ENERGY CONSUMPTION

Figure C.8: Scenarios developed by participants of convergent design session (clockwise): a) *Biosignal in Context*, b) *Processing in Hardware*, c) *Music Augmentation*, and d) *Web Apps*

Appendix D

Outcomes of Music Hack Day UCD action at Sónar 2015

D.1 RAPID-MIX pre-event workshops for Sónar+D 2015 Music Hack Day

D.2 Sónar+D 2015 Music Hack Day groups preliminary survey

1. Group members:
2. Contact:
3. Technologies used:
4. Short description of the hack:
5. Special requirements for the presentation:

D.3 Sónar+D 2015 Music Hack Day follow-up survey

1. Name
2. Age
3. Gender
    - Female
    - Male

Table D.1: RAPID-MIX pre-event workshops for Sónar+D 2015 Music Hack Day

Time	Session Name	Description	Tutor
10.00 10.30	Introduction to Physiological Computing and Biosensing	Physiological data has had a transforming role on multiple aspects of society, which goes beyond the health sciences domains to which they were traditionally associated with. Today, there has been an increasing interest in the use of physiological data for multimedia and creative applications. In this talk we will provide an overview of the different signal sources and their potential applications, focusing on muscle, heart and sympathetic nervous system data.	Hugo Silva
10.30 11.30	BITalino hands-on	In this hands-on session, participants had the chance to experiment with BITalino, a versatile toolkit composed of low-cost hardware and software, created to enable anyone to develop projects and applications involving physiological data. The hardware consists of a modular wireless biosignal acquisition system that can be used to acquire data in real time, interface with other devices (e.g. Arduino or Raspberry PI), or perform rapid prototyping of end-user applications. The software comprehends a set of programming APIs, a biosignal processing toolbox, and a framework for real time data acquisition and post-processing.	Hugo Silva

4. Are you a...
   - Undergraduate student
   - Master Student
   - PhD Student
   - Professional developer
   - Hobbyist
5. What are your programming skills?
   - None
   - Novice
   - Intermediate
   - Advanced
   - Expert
6. Why are you participating in the MHD?
7. Have you been in a hackathon before?

Table D.3: RAPID-MIX pre-event workshops for Sónar+D 2015 Music Hack Day (continuation)

Time	Session Name	Description	Tutor
12.00 12.30	Introduction to interactive machine learning	In this session, participants had an overview of what interactive machine learning is and how it can be used to quickly make new real-time interactive systems, such as new musical instruments, sound visualisations, etc. We described at a high level how supervised learning algorithms permit to create models (functions) by supplying data instead of writing code. These models can be used for gesture or activity recognition (e.g., applying the correct label to a performer's pose or muscle state), or they can be used for mapping input sensor data directly to sound synthesis or animation controls, (e.g., continuously controlling a synthesizer pitch using continuous changes in hand position). Interactive machine learning gives participants an easy way to create training datasets for building supervised learning models, and to correct or refine these models through changes to the training data. This process allows for very fast creation of systems that respond to gesture or audio, and for building systems that are accurate and feel natural to use. This process is also very easy to use, and requires no prior machine learning expertise.	Rebecca Fiebrink
12.30 13.30	Wekinator and GVF hands-on (using BITalino data)	In this hands-on session, participants had the chance to use Wekinator and GVF to apply interactive machine learning to create new gestural controllers and musical instruments. We supplied example code in several common music environments (e.g., Max/MSP, PD, ChucK, SuperCollider), which could easily be modified by participants to build their own musical interactions. We also showed how to grab sensor data from Bitalino, touchOSC, and common input devices and use it to drive this software.	Rebecca Fiebrink
14.30 - 15.00	Introduction to sonic interaction design for mobile and wearables	This session included a range of demos exploring applications for mobile sound processing and interaction using wearable tech prototyping equipment and the Maximilian feature extraction/sound design libraries. Maximilian offers real-time signal processing and feature extraction that runs well on mobile devices such as iPads and iPhones, designed to be useful in rapid prototyping of interactive games and music products.	Mick Grierson
15.00 16.00	Maximilian hands-on, JUCE hands-on (using Wekinator/GVF features computed from BITalino data)	Simple projects were made available in a runnable form that function as building blocks to interface physiological interfaces, machine learning and mobile sound processing. Further example code were also distributed to allow more complex hacks to be created.	Mick Grierson

- Yes
- No

8. How did you find out about RAPID-MIX technologies?
   (e.g. Music Hack Day pre-event workshop; MHD presentation; website...)

9. If there were a RAPID-MIX online forum and code sharing community, how likely would you be to use it?

   - Not at all likely
   - Somewhat likely
   - Very likely
   - Other reason:

10. If you would be somewhat likely to use such an online resource, how would you use it? Check all that apply

    - Moderation or other leadership
    - Sharing my projects with others
    - Finding examples from the community to adapt in my own work
    - Contributing to the codebase of RAPID-MIX
    - Teaching
    - Other reason:

11. What communities around creative coding frameworks do you follow or contribute to?
    (e.g. openprocessing.org, OpenFrameworks, Max/MSP forums, ...)

12. Please briefly describe how and why you participate in these communities.
    (e.g., use them to learn, use them to help out other users, use them for inspiration)

13. Are there any final comments you have for the RAPID-MIX team?

Appendix E

Supporting materials for the expert evaluation of the MFEET prototype

E.1 Multimodal feature extraction and exploration toolkit (MFEET)

E.1.1 Technical description of the MFEET prototype

Facilitating exploration and rapid prototyping of features and feature processing techniques for real-time interaction

E.1.1.1 Use Cases

Development of multimodal interfaces for health, gaming, music, other real-time contexts using audio and/or sensor inputs

E.1.1.2 Goals

Motivation: It is often unclear which audio or sensor features will be most useful for a particular type of analysis or control task. Furthermore, it is often unclear which types of feature processing (e.g., smoothing, thresholding, changing of coordinate system, etc.) should be applied for a task.

Goal: Make it easier for system developers to explore alternative features and alternative feature processing methods when building a new real-time system that uses audio and/or sensor inputs.

E.1.1.3 Questions

1. Which common audio and biosignal features should we prioritize making easy to extract, if the target users are developers using the RAPID-API to build real-time interactive systems?
    - How should we expose parameterization of these features (e.g. allowing change of FFT size)?
2. Which types of feature processing should we prioritize making easy to apply (for these same target users)?
3. What form of GUI-based tools are useful for this type of exploration?
    - What type of control should GUI offer?
    - How should such a GUI tool interoperate with other software/hardware? (e.g., OpenSoundControl messages?)
4. What form of programming-level tools are useful for this type of exploration?
    - What type of control should API offer, beyond mere access to raw feature values?
    - (This question is not addressed directly by this prototype, but learning about how to do this with a GUI should inform future prototypes and development of API-level support for feature exploration and processing)

E.1.1.4 Description

E.1.2 MFEET video-prototype voice-off script

The Multimodal Feature Extraction and Exploration Toolkit (MFEET) is a suite of standalone tools that can be connected with each other or with developers' code, which make it possible to experiment with different features and different signal processing techniques very quickly, without programming or worrying about how to interface with hardware. The MFEET assists on the task of determining exactly what information to extract from a raw signal when working with audio signals, physiological signals, or signals from many other types of sensors.

There are many established feature extraction techniques for working with specific types of signals, but even people who are experts in signal processing may not know what techniques will be best for a particular application. Furthermore, someone sitting down to work on a new creative project may not even know what exactly he or she wants to build! They may have to experiment with many types of feature extractors (and even many types of sensors) before deciding what type of real-time interaction or analysis they want to support.

The first component can be used to visualise the data extracted from several different sensors on screen—e.g. BITalino sensors (Guerreiro et al., 2013); this visualisation helps users to understand what is being sensed, as they make connections between their interactions with sensors and the motions on the screen. It also helps people to verify that BITalino and its sensors are all connected properly. A developer can use the on-screen checkboxes to select features they want to use; these features are then sent out via a simple communication protocol called OSC. People using audio input can also visualise of a number of audio analysis features that are useful in analysing pitch, timbre, voice, instrumentation, and so on. Again, this program sends out OSC data for all the selected audio analysis features.

A developer can write their program in any language to receive these OSC values, or they could use Wekinator to apply machine learning to these values, perhaps training a gesture or audio classifier. However, the developer never has to worry about how to write code to communicate with the hardware, and they do not have to read up on how to extract meaningful information from BITalino or an audio stream. They can use this program to experiment with different features and be up and running in a matter of seconds.

A second standalone prototype application built as part of the MFEET allows users to apply other processing to real-time data streams. This helps developers who still need to do further computation on the features extracted from sensors or audio, which might be useful for many types of applications (again, without having to write any code). For example, if one is using the BITalino accelerometer to detect a shake, the current accelerometer value is often not the actual information of interest; it might be more interesting to have features like the maximum accelerometer value that has been seen over the past few hundred milliseconds. One can add this extra processing using the GUI, like this. Or, if using the BITalino EMG sensor, one might want to add a smoothing filter that reduces the noise in my signal. A very simple filter can be created (e.g. by taking the average of values over some time window) or one can type in a more sophisticated digital filter as an equation obtained from a filter design tool (e.g.). As easily seen, this feature processing application provides a set of common processing techniques, as well as the ability to type in arbitrary mathematical expressions.

This tool also provides users with the ability to change the rate at which data is processed and sent, and to apply simple thresholds or other tests. Users can also monitor both the original incoming signal data and the modified signal values. This feature-processing tool receives real-time data from any source using OSC. This data could come from the BITalino, audio feature extraction tools previously described, or from another user-created program. For this prototype, we have built three OSC bridges: one for audio feature extraction, one for sourcing the EMG and the accelerometer from BITalino

> *and a Myo OSC bridge that sources both the 8 EMG channels and the IMU data. For instance, in Figure 10, the Weki Input helper, on the right, receives data from the BITalino extractor, on the left. Furthermore, this program sends out the modified feature values using OSC; again, this could be sent to any user-created program in any language, or it could be sent to Wekinator for machine learning. Here, it's being sent to Wekinator, where we're using it to control sound.*
>
> (R. Fiebrink, 2017)

E.2 Experimental procedure script using walktrough and wizard-of-Oz

1. Briefing about the procedure and introduction to the MFEET prototype (10 minutes)
   (a) Bridge – Max/MSP patch (BITalino, Myo—EMG and motion data)
   (b) Weki Input Helper
   (c) Wekinator
   (d) Sink – Max/MSP patch (sonification)
2. Setting up the pipeline for prototyping signal processing (10 minutes).
3. Applying features for classification and gesture recognition (20 minutes).
   (a) Average over a sliding window
   (b) Threshold detection
   (c) Mathematical filter expression
4. Eliciting the identification of the space of compelling applications and tasks which MFEET could be applied for as an easy-to-use development tool (10 minutes).
5. Eliciting understanding about which features should be prioritised for making easy to extract and to apply in the the previously identified development contexts (10 minutes).
6. Eliciting the perception about of the usability of MFEET, concerning the exploration, configuration and state of the pipeline and feature extractors (10 minutes).

Appendix F

Personal communication excerpts from RAPID-MIX stakeholders

F.1 R. Fiebrink internal communication to Goldsmiths team on 2016/02/17

Had a good meeting with Fred, Joseph, and Norbert today to talk about RAPID API for ML. Fred will send out a summary email to all partners soon. In the meantime, here are the high-level points:

- *We believe that RAPID API for ML is most clearly distinguished from other ML APIs and Frameworks because of (1) needing to support fluid, easy interactions across multiple devices, (2) needing to support streaming real-time data, (3) needing to support end-user curation and modification of training examples. In other words, it's about supporting interaction between user & algorithm, and across multiple devices including mobile & Web, rather than about algorithms. We may even want to provide hooks between our framework and existing API/frameworks like e.g. tensor flow. They are complementary.*
- *IRCAM's work with Orbe to build a user context classifier is going to be one of a few use cases we can rely on to drive the tools & data representations that become the RAPID API for ML. In a nutshell: We want a user (developer) to be able to capture sensor data "in the wild" using a mobile/wearable, upload that to a server (in realtime or asynchronously), use a web-based toolset to interact with data on server to visualise, segment, label, train and configure algorithms, etc. (i.e., do the most important stuff that users of RepoVizz, Wekinator, etc. are doing), store data longterm on server and also aggregate data from multiple users/data sources, and deploy trained models back to mobile/embedded devices. Web frameworks &*

JSON can provide some glue to hold these pieces together. We may re-implement some of the existing functionality of RepoVizz/Wekinator/etc. in web tools, and make data in formats that can be imported/exported from those tools, but also make it possible for others to build other interfaces on top of our data/APIs here. In other words, we might have reference implementations of user interfaces, but we'll mainly be focused on the data formats, communication that ties these components together and makes it easier for lots of different types of interfaces to be built on top.

- *It is likely that many of our other project—whether within RAPID-MIX formally or not— will require a very similar workflow, so our independent progress on them will all inform the shape of the RAPID API tools. For instance, the work I'm doing with micro:bit requires a similar workflow. I imagine Atau's EMG sensing may look similar. So will some of the MIX products. Let's make a note of what all these projects are so we can think about how people would architect similar systems more easily if they had the tools we're about to make.*
- *We discussed the PIPO framework in a lot more detail, and I think it's going to be very useful for both feature extractors and machine learning components. In a nutshell, PIPO gives us a very lightweight starting point for an "API" for signal processing functions that need to be chained together and scheduled, as well as several reference implementations of "host" environments that can schedule and run PIPO chains in realtime. These chains could be run in any number of environments (from C++ code on embedded systems to JavaScript to Max/MSP). Longer term, they could even be the foundation for a visual dataflow-like programming interface for signal processing & ML design for RAPID API. Inserting trained models into PIPO wrappers would allow models to be run together in parallel and in hierarchies, without needing to write extra code to manage dataflow/scheduling/etc. Joseph has already packaged some Maximilian feature extractors within PIPO. A lot of the "input helper" functions that I've written recently are also already available as PIPO objects, and it's easy for us to take the ones that aren't and wrap them in PIPO as well. These plus maximilian (plus perhaps essentia features?) can be put in our Gitlab repository and immediately be useful to all project partners, even before we have any other RAPID API infrastructure built. We can take Michael's C++ model functions and wrap them in PIPO very easily. Diemo is still evolving PIPO and is eager to make modifications if they are needed to support what we want to do.*

F.2 M. Zbyszyński's internal communication early development of RapidLib

The initial problem I was trying to solve is this: "As a Wekinator user, I want to be able to take a model I just trained in Wekinator and run it on my embedded processor." Outputting hard-coded C++ models was the solution to this. These models have a minimal memory footprint. They have no training features or exposed parameters. These models are minimal, and don't use any C++11 or fancy code that might not run on older processors/compilers. This is one prototype-to-production pathway.

I started with this problem because it was the simplest, and would lead to code that I could expand upon for other solutions, like:

"As a Wekinator user, I want to be able take a model I just trained in Wekinator and run it in a web browser." This is where Emscripten came in. Emscripten was Mick's suggestion, and it made sense to maintain one (C++) codebase rather than two. I did actually code some algorithms in native JS before I got this working. Also, Emscripten prompted me to use more modern C++; classes like std::vector work better when ported to JS.

"I want to be able to move my trained models from one place (Wekinator, C++ app, web app,. . .) to someplace else. Or, save them and load them in a future project." JSON import/export is the solution to all of the RAPID-MIX saving, moving and loading use cases. We chose JSON because it's modern and familiar, native to JavaScript, built in to Java, and already present in XMM. I worked on the JSON solution later because it involved more design (agreeing on schemas) and more coding.

Maybe some day Wekinator will import RAPID-MIX JSON.

mz

Appendix G

RAPID-MIX API related data

G.1 Listing of M. Zbyszyński's commits history to his Wekinator (wekimini) fork

77b2a68 Michael Zbyszynski 2016-01-06 Adding code to write cpp source for model

1c2129f Michael Zbyszynski 2016-01-07 Pulling some model parameters from the model. Still need numNeibours and numClasses.

5e98e53 Michael Zbyszynski 2016-01-13 All parameters in the header file are now coming from wekinator

3a558a9 Michael Zbyszynski 2016-01-13 produces C++ code that builds and runs

487b9a5 Michael Zbyszynski 2016-01-13 fixed bug constructing neighbours object

b50c66f Michael Zbyszynski 2016-02-02 Merge remote-tracking branch 'upstream/master'

1241e86 Michael Zbyszynski 2016-02-05 Merge remote-tracking branch 'upstream/master'

cd2bf75 Michael Zbyszynski 2016-02-05 roughing in neuralNetwork Cpp code

3482a38 mzed 2016-02-09 writes a dummy NN model. still needs real data.

f46ac65 Michael Zbyszynski 2016-02-09 Merge remote-tracking branch 'upstream/master'

8045522 mzed 2016-02-09 Nearly working c++ models. Need to verify model format.

28b496f mzed 2016-02-09 fixed use of bias nodes

c5e04c4 mzed 2016-02-10 possibly working, except for bug getting number of hidden nodes

e505fc0 mzed 2016-02-10 fixed bug in wHiddenOutput ordering

2f72b2d mzed 2016-02-10 fixed problem getting number of hidden nodes

0fba9ff mzed 2016-02-11 optimized based on single output per model

64c9440 mzed 2016-02-22 Merge remote-tracking branch 'upstream/master'

fa0e42a Michael Zbyszynski 2016-03-08 Merge remote-tracking branch 'upstream/master'

1954e45 mzed 2016-03-08 knn::getClass now takes a pointer to an array, rather than a struct

3b7f31e mzed 2016-03-16 Merge remote-tracking branch 'upstream/master'

3b35842 mzed 2016-03-16 changed include to grab model-0

```
648c14c mzed 2016-03-17 more efficient writing and instantiation of NN models. work in progress
09281fa mzed 2016-03-18 Refactoring export of cpp model
c3fdc56 mzed 2016-03-19 For NN, output Cpp is pretty close to the original model :-)
5956136 mzed 2016-03-19 clipping function from weka
4ff13c9 mzed 2016-03-21 writing values for input normalization
90ef239 mzed 2016-04-06 Properly normalizing output
19891dc mzed 2016-04-06 Merge remote-tracking branch 'upstream/master'
1025e74 mzed 2016-04-12 Generalized for multiple models with different inputs
e2d1045 mzed 2016-04-12 Merge remote-tracking branch 'upstream/master'
f9a33d8 mzed 2016-04-18 Selecting proper inputs
5087a43 mzed 2016-04-18 Selecting input names properly
2f4fb87 mzed 2016-04-18 Handling multiple hidden layers for NN
c5fae40 mzed 2016-04-19 (tag: v0.1-beta) Updated kNN to work like NN
4a2c77f mzed 2016-05-27 (tag: v0.2-beta) First stab at modelSet implementation
dcd566d mzed 2016-05-31 fixed errors in kNN export
8d07fe5 mzed 2016-06-01 (tag: v0.3-beta) Moved setup to wekiModelSet class; fixes #1 and #2
616b0e0 mzed 2016-06-01 fixes #2, improved kNN
c3755e5 mzed 2016-06-03 (tag: v0.4-beta) fixed bug in kNN selection
3e75d49 mzed 2016-06-24 (tag: v0.5-beta) Cleaning up destructors
3fc43a6 mzed 2016-07-13 Merge remote-tracking branch 'upstream/master'
925c12a mzed 2016-07-13 small typos in C++ export
a2dcc52 mzed 2016-07-13 not including classes that aren't exported
d686083 mzed 2016-07-13 (tag: v0.6-beta) No longer declaring variables for unused algorithms
aaab247 mzed 2016-08-26 Now writing JSON for NN models
872bdeb mzed 2016-08-26 (tag: v0.7-beta) Writes JSON for kNN and NN models
1cfb896 mzed 2016-08-26 Proper file extension, proper knn model type, Fixes #4
1ea6931 mzed 2016-09-05 (tag: v0.8-beta) Added in/out Mins and Maxes to JSON
e651862 mzed 2016-10-13 This looks like a nice change. I'm not sure why I did it.
f4f81a4 mzed 2016-11-17 (tag: v0.9-beta) Changed JSON to output in—out ranges—bases, which is like the C++ API.
164faaa mzed 2016-11-17 Got the maths right for ranges and bases
7a23d4e mzed 2017-01-05 Fixes #5 Properly handles input names with spaces
b89264d mzed 2017-01-05 (tag: v0.9.1-beta) a bit of a lame fix. Fixes #6
1dd10e8 mzed 2017-01-11 (HEAD -¿ master, origin/master, origin/HEAD) Merge remote-tracking branch 'upstream/master'
```

This commit history is accessible from the remote git repository at https://github.com/mzed/wekimini, with the following command:

```
git log –author="\(mzed\)\—\(Michael\)" —oneline —branches –reverse –pretty=format:"%h
```

%an %ad %x09 %d %x20 %s" –date=short

G.2 M. Zbyszyński's internal communication about early RapidLib user stories

The first thing I did was modify Wekinator to output a hard-coded kNN model. The implementation Weka fairly complex, so I implemented all of the ML algorithms from scratch, or based on looking at other C++ examples.

The initial problem I was trying to solve is this: "As a Wekinator user, I want to be able to take a model I just trained in Wekinator and run it on my embedded processor." Outputting hard-coded C++ models was the solution to this. These models have a minimal memory footprint. They have no training features or exposed parameters. These models are minimal, and don't use any C++11 or fancy code that might not run on older processors/compilers. This is one prototype-to-production pathway.

I started with this problem because it was the simplest, and would lead to code that I could expand upon for other solutions, like: "As a Wekinator user, I want to be able take a model I just trained in Wekinator and run it in a web browser." This is where Emscripten came in. Emscripten was Mick's suggestion, and it made sense to maintain one (C++) codebase rather than two. I did actually code some algorithms in native JS before I got this working. Also, Emscripten prompted me to use more modern C++; classes like std::vector work better when ported to JS.

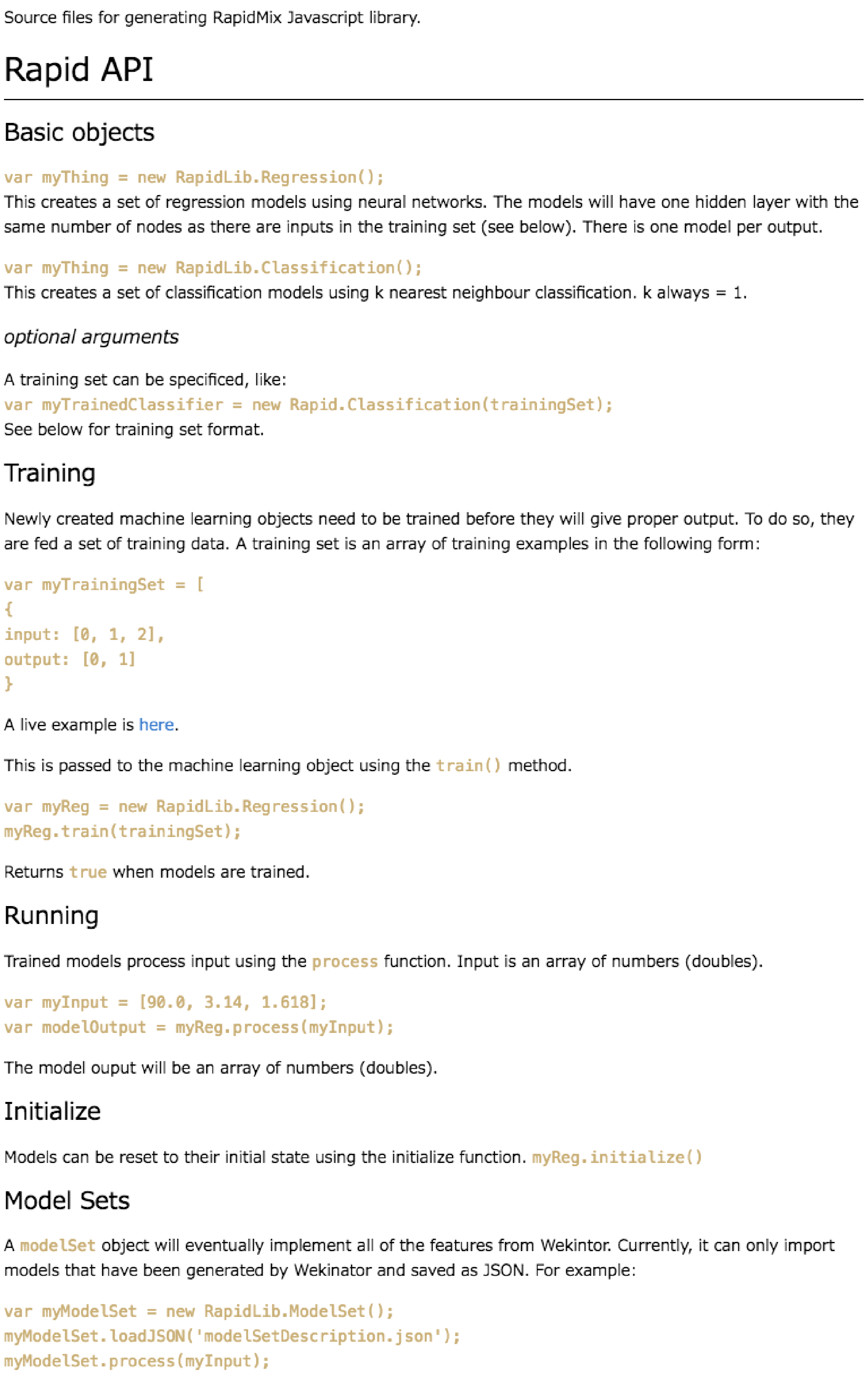
"I want to be able to move my trained models from one place (Wekinator, C++ app, web app, . . .) to someplace else. Or, save them and load them in a future project." JSON import/export is the solution to all of the RAPID-MIX saving, moving and loading use cases. We chose JSON because it's modern and familiar, native to JavaScript, built in to Java, and already present in XMM. I worked on the JSON solution later because it involved more design (agreeing on schemas) and more coding.

Maybe some day Wekinator will import RAPID-MIX JSON.

mz

G.3 README for RapidLib.js v1.0

G.4 J. Larralde's UML diagram of the RAPID-MIX API XMM C++ subset



Source files for generating RapidMix Javascript library.

Rapid API

Basic objects

```
var myThing = new RapidLib.Regression();
```

This creates a set of regression models using neural networks. The models will have one hidden layer with the same number of nodes as there are inputs in the training set (see below). There is one model per output.

```
var myThing = new RapidLib.Classification();
```

This creates a set of classification models using k nearest neighbour classification. k always = 1.

optional arguments

A training set can be specificed, like:

```
var myTrainedClassifier = new Rapid.Classification(trainingSet);
```

See below for training set format.

Training

Newly created machine learning objects need to be trained before they will give proper output. To do so, they are fed a set of training data. A training set is an array of training examples in the following form:

```
var myTrainingSet = [
{
input: [0, 1, 2],
output: [0, 1]
}
```

A live example is here.

This is passed to the machine learning object using the `train()` method.

```
var myReg = new RapidLib.Regression();
myReg.train(trainingSet);
```

Returns `true` when models are trained.

Running

Trained models process input using the `process` function. Input is an array of numbers (doubles).

```
var myInput = [90.0, 3.14, 1.618];
var modelOutput = myReg.process(myInput);
```

The model ouput will be an array of numbers (doubles).

Initialize

Models can be reset to their initial state using the initialize function. `myReg.initialize()`

Model Sets

A `modelSet` object will eventually implement all of the features from Wekintor. Currently, it can only import models that have been generated by Wekinator and saved as JSON. For example:

```
var myModelSet = new RapidLib.ModelSet();
myModelSet.loadJSON('modelSetDescription.json');
myModelSet.process(myInput);
```

Figure G.1: Readme for RapidLib.js

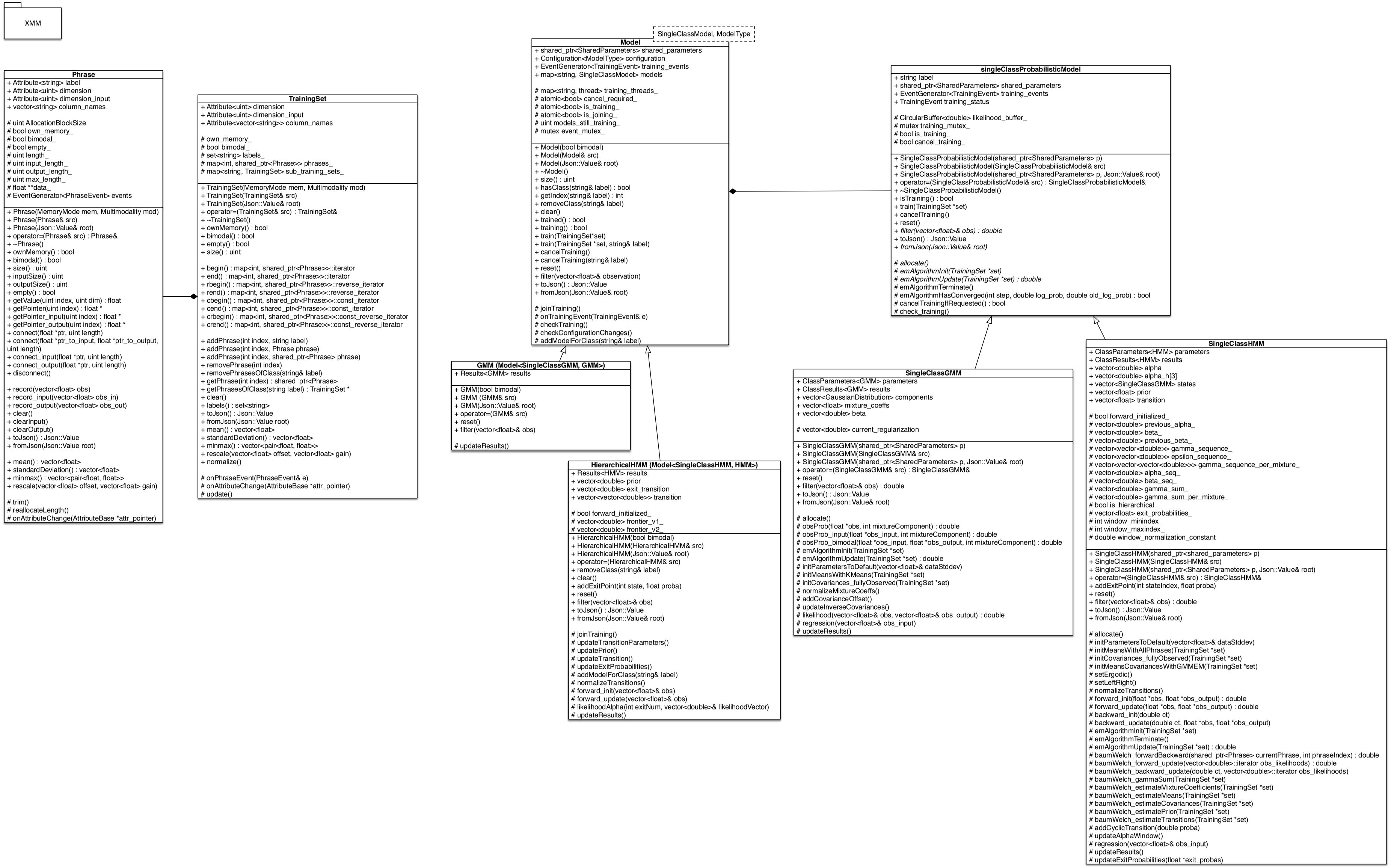

XMM
Phrase
+ Attribute<string> label
+ Attribute<uint> dimension
+ Attribute<uint> dimension_input
+ vector<string> column_names
# uint AllocationBlockSize
# bool own_memory_
# bool bimodal_
# bool empty_
# uint length_
# uint input_length_
# uint output_length_
# uint max_length_
# float **data_
# EventGenerator<PhraseEvent> events
+ Phrase(MemoryMode mem, Multimodality mod)
+ Phrase(Phrase& src)
+ Phrase(Json::Value& root)
+ operator=(Phrase& src) : Phrase&
+ ~Phrase()
+ ownMemory() : bool
+ bimodal() : bool
+ size() : uint
+ inputSize() : uint
+ outputSize() : uint
+ empty() : bool
+ getValue(uint index, uint dim) : float
+ getPointer(uint index) : float *
+ getPointer_input(uint index) : float *
+ getPointer_output(uint index) : float *
+ connect(float *ptr, uint length)
+ connect(float *ptr_to_input, float *ptr_to_output, uint length)
+ connect_input(float *ptr, uint length)
+ connect_output(float *ptr, uint length)
+ disconnect()
+ record(vector<float> obs)
+ record_input(vector<float> obs_in)
+ record_output(vector<float> obs_out)
+ clear()
+ clearInput()
+ clearOutput()
+ toJson() : Json::Value
+ fromJson(Json::Value root)
+ mean() : vector<float>
+ standardDeviation() : vector<float>
+ minmax() : vector<pair<float, float>>
+ rescale(vector<float> offset, vector<float> gain)
# trim()
# reallocateLength()
# onAttributeChange(AttributeBase *attr_pointer)
TrainingSet
+ Attribute<uint> dimension
+ Attribute<uint> dimension_input
+ Attribute<vector<string>> column_names
# own_memory_
# bool bimodal_
# set<string> labels_
# map<int, shared_ptr<Phrase>> phrases_
# map<string, TrainingSet> sub_training_sets_
+ TrainingSet(MemoryMode mem, Multimodality mod)
+ TrainingSet(TrainingSet& src)
+ TrainingSet(Json::Value& root)
+ operator=(TrainingSet& src) : TrainingSet&
+ ~TrainingSet()
+ ownMemory() : bool
+ bimodal() : bool
+ empty() : bool
+ size() : uint
+ begin() : map<int, shared_ptr<Phrase>>::iterator
+ end() : map<int, shared_ptr<Phrase>>::iterator
+ rbegin() : map<int, shared_ptr<Phrase>>::reverse_iterator
+ rend() : map<int, shared_ptr<Phrase>>::reverse_iterator
+ cbegin() : map<int, shared_ptr<Phrase>>::const_iterator
+ cend() : map<int, shared_ptr<Phrase>>::const_iterator
+ crbegin() : map<int, shared_ptr<Phrase>>::const_reverse_iterator
+ crend() : map<int, shared_ptr<Phrase>>::const_reverse_iterator
+ addPhrase(int index, string label)
+ addPhrase(int index, Phrase phrase)
+ addPhrase(int index, shared_ptr<Phrase> phrase)
+ removePhrase(int index)
+ removePhrasesOfClass(string& label)
+ getPhrase(int index) : shared_ptr<Phrase>
+ getPhrasesOfClass(string label) : TrainingSet *
+ clear()
+ labels() : set<string>
+ toJson() : Json::Value
+ fromJson(Json::Value root)
+ mean() : vector<float>
+ standardDeviation() : vector<float>
+ minmax() : vector<pair<float, float>>
+ rescale(vector<float> offset, vector<float> gain)
+ normalize()
# onPhraseEvent(PhraseEvent& e)
# onAttributeChange(AttributeBase *attr_pointer)
# update()
SingleClassModel, ModelType
Model
+ shared_ptr<SharedParameters> shared_parameters
+ Configuration<ModelType> configuration
+ EventGenerator<TrainingEvent> training_events
+ map<string, SingleClassModel> models
# map<string, thread> training_threads_
# atomic<bool> cancel_required_
# atomic<bool> is_training_
# atomic<bool> is_joining_
# uint models_still_training_
# mutex event_mutex_
+ Model(bool bimodal)
+ Model(Model& src)
+ Model(Json::Value& root)
+ ~Model()
+ size() : uint
+ hasClass(string& label) : bool
+ getIndex(string& label) : int
+ removeClass(string& label)
+ clear()
+ trained() : bool
+ training() : bool
+ train(TrainingSet*set)
+ train(TrainingSet *set, string& label)
+ cancelTraining()
+ cancelTraining(string& label)
+ reset()
+ filter(vector<float>& observation)
+ toJson() : Json::Value
+ fromJson(Json::Value& root)
# joinTraining()
# onTrainingEvent(TrainingEvent& e)
# checkTraining()
# checkConfigurationChanges()
# addModelForClass(string& label)
GMM (Model<SingleClassGMM, GMM>)
+ Results<GMM> results
+ GMM(bool bimodal)
+ GMM (GMM& src)
+ GMM(Json::Value& root)
+ operator=(GMM& src)
+ reset()
+ filter(vector<float>& obs)
# updateResults()
HierarchicalHMM (Model<SingleClassHMM, HMM>)
+ Results<HMM> results
+ vector<double> prior
+ vector<double> exit_transition
+ vector<vector<double>> transition
# bool forward_initialized_
# vector<double> frontier_v1_
# vector<double> frontier_v2_
+ HierarchicalHMM(bool bimodal)
+ HierarchicalHMM(HierarchicalHMM& src)
+ HierarchicalHMM(Json::Value& root)
+ operator=(HierarchicalHMM& src)
+ removeClass(string& label)
+ clear()
+ addExitPoint(int state, float proba)
+ reset()
+ filter(vector<float>& obs)
+ toJson() : Json::Value
+ fromJson(Json::Value& root)
# joinTraining()
# updateTransitionParameters()
# updatePrior()
# updateTransition()
# updateExitProbabilities()
# addModelForClass(string& label)
# normalizeTransitions()
# forward_init(vector<float>& obs)
# forward_update(vector<float>& obs)
# likelihoodAlpha(int exitNum, vector<double>& likelihoodVector)
# updateResults()
singleClassProbabilisticModel
+ string label
+ shared_ptr<SharedParameters> shared_parameters
+ EventGenerator<TrainingEvent> training_events
+ TrainingEvent training_status
# CircularBuffer<double> likelihood_buffer_
# mutex training_mutex_
# bool is_training_
# bool cancel_training_
+ SingleClassProbabilisticModel(shared_ptr<SharedParameters> p)
+ SingleClassProbabilisticModel(SingleClassProbabilisticModel& src)
+ SingleClassProbabilisticModel(shared_ptr<SharedParameters> p, Json::Value& root)
+ operator=(SingleClassProbabilisticModel& src) : SingleClassProbabilisticModel&
+ ~SingleClassProbabilisticModel()
+ isTraining() : bool
+ train(TrainingSet *set)
+ cancelTraining()
+ reset()
+ *filter(vector<float>& obs) : double*
+ toJson() : Json::Value
+ *fromJson(Json::Value& root)*
# allocate()
*# emAlgorithmInit(TrainingSet *set)*
*# emAlgorithmUpdate(TrainingSet *set) : double*
# emAlgorithmTerminate()
# emAlgorithmHasConverged(int step, double log_prob, double old_log_prob) : bool
# cancelTrainingIfRequested() : bool
# check_training()
SingleClassGMM
+ ClassParameters<GMM> parameters
+ ClassResults<GMM> results
+ vector<GaussianDistribution> components
+ vector<float> mixture_coeffs
+ vector<double> beta
# vector<double> current_regularization
+ SingleClassGMM(shared_ptr<SharedParameters> p)
+ SingleClassGMM(SingleClassGMM& src)
+ SingleClassGMM(shared_ptr<SharedParameters> p, Json::Value& root)
+ operator=(SingleClassGMM& src) : SingleClassGMM&
+ reset()
+ filter(vector<float>& obs) : double
+ toJson() : Json::Value
+ fromJson(Json::Value& root)
# allocate()
# obsProb(float *obs, int mixtureComponent) : double
# obsProb_input(float *obs_input, int mixtureComponent) : double
# obsProb_bimodal(float *obs_input, float *obs_output, int mixtureComponent) : double
# emAlgorithmInit(TrainingSet *set)
# emAlgorithmUpdate(TrainingSet *set) : double
# initParametersToDefault(vector<float>& dataStddev)
# initMeansWithKMeans(TrainingSet *set)
# initCovariances_fullyObserved(TrainingSet *set)
# normalizeMixtureCoeffs()
# addCovarianceOffset()
# updateInverseCovariances()
# likelihood(vector<float>& obs, vector<float>& obs_output) : double
# regression(vector<float>& obs_input)
# updateResults()
SingleClassHMM
+ ClassParameters<HMM> parameters
+ ClassResults<HMM> results
+ vector<double> alpha
+ vector<double> alpha_h[3]
+ vector<SingleClassGMM> states
+ vector<float> prior
+ vector<float> transition
# bool forward_initialized_
# vector<double> previous_alpha_
# vector<double> beta_
# vector<double> previous_beta_
# vector<vector<double>> gamma_sequence_
# vector<vector<double>> epsilon_sequence_
# vector<vector<vector<double>>> gamma_sequence_per_mixture_
# vector<double> alpha_seq_
# vector<double> beta_seq_
# vector<double> gamma_sum_
# vector<double> gamma_sum_per_mixture_
# bool is_hierarchical_
# vector<float> exit_probabilities_
# int window_minindex_
# int window_maxindex_
# double window_normalization_constant
+ SingleClassHMM(shared_ptr<shared_parameters> p)
+ SingleClassHMM(SingleClassHMM& src)
+ SingleClassHMM(shared_ptr<SharedParameters> p, Json::Value& root)
+ operator=(SingleClassHMM& src) : SingleClassHMM&
+ addExitPoint(int stateIndex, float proba)
+ reset()
+ filter(vector<float>& obs) : double
+ toJson() : Json::Value
+ fromJson(Json::Value& root)
# allocate()
# initParametersToDefault(vector<float>& dataStddev)
# initMeansWithAllPhrases(TrainingSet *set)
# initCovariances_fullyObserved(TrainingSet *set)
# initMeansCovariancesWithGMMEM(TrainingSet *set)
# setErgodic()
# setLeftRight()
# normalizeTransitions()
# forward_init(float *obs, float *obs_output) : double
# forward_update(float *obs, float *obs_output) : double
# backward_init(double ct)
# backward_update(double ct, float *obs, float *obs_output)
# emAlgorithmInit(TrainingSet *set)
# emAlgorithmTerminate()
# emAlgorithmUpdate(TrainingSet *set) : double
# baumWelch_forwardBackward(shared_ptr<Phrase> currentPhrase, int phraseIndex) : double
# baumWelch_forward_update(vector<double>::iterator obs_likelihoods) : double
# baumWelch_backward_update(double ct, vector<double>::iterator obs_likelihoods)
# baumWelch_gammaSum(TrainingSet *set)
# baumWelch_estimateMixtureCoefficients(TrainingSet *set)
# baumWelch_estimateMeans(TrainingSet *set)
# baumWelch_estimateCovariances(TrainingSet *set)
# baumWelch_estimatePrior(TrainingSet *set)
# baumWelch_estimateTransitions(TrainingSet *set)
# addCyclicTransition(double proba)
# updateAlphaWindow()
# regression(vector<float>& obs_input)
# updateResults()
# updateExitProbabilities(float *exit_probas)

Figure G.2: UML Class diagram of RapidLib

Appendix H

Cognitive Dimensions Questionnaire

Q1. How appropriate have you found the overall API abstraction level to be for your development needs? How would you classify the abstraction level of the types of classes and methods that you had to work with,

□ too high level — □ too low level — □ just right

Why?

Q2. Did you feel that you had to understand the underlying implementation to be able to use the functionalities of the API? Why?

Q3. What previous Machine Learning (ML) knowledge made it easy to use the API?

Q4. What other areas of knowledge (other than ML) would have made it easier to use the API?

Q5. What knowledge about the API components was essential to achieve your development and implementation goals?

Q6. How did you go about learning how to use the API? (Check all that apply)

□ I wrote a couple of lines of code to try to get something working and built up understanding from that

□ I copied sample code provided with the API

□ I read a high level overview of the API first; I only started writing code once I had a general idea about the architecture of the API

□ I learned about the API components and their dependencies before starting to do anything useful related to my development goal

□ I learned about the underlying ML architecture of the API and other conceptual information (e.g., specific algorithms, different approaches, etc) before starting to do anything useful related to my development goal.

Q7. Can you give examples of objects with unclear dependencies between them?

Q8. How would you describe your experiences in learning how to use the API? Was there

□ too much to learn — □ not enough to learn — □ just right

Why?

Q9. How did the different API documentation elements supported the learning that you needed to progress with your development goals?

a) Website

□ unsatisfactory — □ improvement needed — □ good — □ very good — □ outstanding — □ did not use

b) Git repository and README file

□ unsatisfactory — □ improvement needed — □ good — □ very good — □ outstanding — □ did not use

c) Code examples

□ unsatisfactory — □ improvement needed — □ good — □ very good — □ outstanding — □ did not use

d) Code comments

□ unsatisfactory — □ improvement needed — □ good — □ very good — □ outstanding — □ did not use

Q10. What 'context' did you have to keep in mind in order to use the API functionalities you needed for your development goals? (Check all that apply)

□ type(s) of ML algorithm

□ data structure(s) for training and model input/output

□ local scope variables

□ global scope variables

□ registered events

□ API classes

□ API methods

□ other components

□ app configuration settings

□ system configuration settings

□ database (e.g., JSON, XML, CSV, database management service)

□ other (please describe):

Q11. How would you describe your experiences with respect to the amount of 'context' that you had to keep in mind to implement what you wanted?

□ too much to learn — □ not enough to learn — □ just right

Why?

Q12. How would you describe your experience with respect to the overall amount of code you had to write to use the API?

Q13. Were you able to evaluate your progress in using the API when you needed to, or did it require more work than expected to evaluate progress? Why?

Q14. When you were programming with the API, were you able go about the process in any order you liked or were you forced to think ahead and make certain decisions first?

Q15. What main decisions did you need to make in advance, in order to achieve your development goals?

Q16. Was it obvious that you needed to make these decisions? Did you learn about this through trial and error?

Q17. What sort of problems did this cause in your work?

Q18. How easy is it to see or find the details of the API while you are using it? What kinds of things are more difficult to see or find? Why?

Q19. Did the API and its documentation provide enough information about the relevant machine learning specifics related to your development goal?

Q20. What information was missing, or that you had to find by referring to external sources?

Q21. How much of the underlying details of the machine learning did you have to understand in order to be able to use the API successfully? How much of those details are exposed in the API?

Q22. By what process did you understand the intricate working details of the API while you worked on your implementation?

Q23. Were you able to find enough information to understand the intricate working details of the API while you work on your programming task? If not, what are the information you think is missing?

Q24. Were you able to use the API exactly 'as-is' or did you feel the need to adapt the API to meet your needs? Did you (check all that apply):

□ fork the API?

□ build any adapter class(es) for using it?

□ derive or extend any classes or types?

□ override methods?

□ hack the API?

□ any other?

Why?

Q25. When you needed to make changes to code that you had previously written using the API, how easy was it to make these changes? Why?

Q26. If you noticed elements of the API that offered similar functionalities? Were the differences between them made clear to you? Please explain.

Q27. When reading code that uses the API, is it easy to tell what each section of code does?

□ yes — □ no

Q28. Can you give examples of parts that were particularly difficult to interpret?

□ yes — □ no

Q29. Was it easy to know which API classes and methods to use when writing code?

□ yes — □ no

Q30. When using the API, do the exposed classes and methods map easily to the conceptual objects that you think about manipulating?

□ yes — □ no

Q31. Can you give examples of any such conceptual objects that map easily? Or that do not?

Q32. When using the API, was it easy to map from your conceptual ideas and application functionalities to API code?

□ yes — □ no

Why?

Q33. Were there any incidents in which you used the API incorrectly?

Q34. Did the API offer any help to identify that you misused it? If any similar incidents occurred, can you please explain.

Q35. Did you eventually identify the correct way of using it?

Q36. Did the API give proper error messages in case of exceptions or error? Did you have to handle them at application level? If you had to handle them at application level, please mention the scenarios.

Q37. Did you use any of the following testing or evaluation as your application developed or after you completed it?

a) Objective evaluation with unit tests and fixtures:

□ integration or functional test (higher level test for application areas wrapping API features)

□ seam tests for unstable data (unit tests for validation of model tolerance between data input and outputs)

□ training accuracy test

□ cross validation tests of model fit

□ precision/recall

b) Subjective evaluation with direct tests and direct observation of model behaviour:

□ Correctness (percentage of test cases where model produces the desired result)

□ Cost (weighted correctness; do the most important test cases produce the desired result)

□ Decision boundary (does a classifier change label in the right place)

□ Confidence (is the model more confident when it produces correct outputs for easy problems, and less confident when it produces outputs for hard problems, or produces incorrect outputs)

□ Other criteria (e.g., complexity, expressive potential, unexpectedness, ease of use) — please describe:

Q38. Why did you test these things?

Q39. Did the API provided any guidance on how to test your application? Did you get guidance from elsewhere?

Q40. Do you find yourself using the API in ways that seemed unusual, or ways that the implementers might not have intended? Can you give some examples?