UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DEVELOPMENT OF AN EMOTION AWARE AMBIENT INTELLIGENT SYSTEM FOR SMART OFFICE

APPLICATION IN A LIVING LAB AND IN A SOCIAL SIMULATED SCENARIO

SERGIO MUÑOZ LÓPEZ

2017

PROYECTO FIN DE CARRERA

Título:	Desarrollo de un sistema inteligente con reconocimiento de emociones para oficinas inteligentes. Aplicación en un laboratorio real y en un escenario simulado
Título (inglés):	Development of an emotion aware ambient intelligent system for smart offices. Application in a living lab and in a social simulated scenario
Autor:	Sergio Muñoz López
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	-
Vocal:	-
Secretario:	-
Suplente:	-

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE MÁSTER

DEVELOPMENT OF AN EMOTION AWARE AMBIENT INTELLIGENT SYSTEM FOR SMART OFFICES

APPLICATION IN A LIVING LAB AND IN A SOCIAL SIMULATED SCENARIO

Sergio Muñoz López

Junio de 2017

Resumen

El éxito de las plataformas de dispositivos móviles inteligentes ha promovido la evolución de la Internet de las Cosas (IoT) desde una lejana visión futurística a una alcanzable realidad. Estos dispositivos proporcionan valiosos datos provenientes del comportamiento humano, del ambiente o de la infraestructura IoT, habilitando un amplio abanico de servicios y aplicaciones dependientes del entorno.

Sin embargo, a pesar de la proliferación del IoT, y aunque se ha investigado mucho en el impacto que las emociones tienen en distintos dominios (como la salud, las relaciones sociales, la vida laboral o casos de emergencia), el número de entornos que tienen en cuenta el comportamiento emocional es aún pequeño. Este trabajo de fin de máster tiene como objetivo salvar esta distancia, y propone el desarrollo de una plataforma IoT que reconozca emociones, cruce los datos con datos relativos al contexto, y personalice el entorno del usuario de acuerdo a estos datos.

El desarrollo de esta plataforma es el principal objetivo de este trabajo. El sistema inteligente desarrollado debe ser capaz de capturar emociones a través del análisis de datos provenientes de cámaras y diferentes sensores, y automáticamente adaptar el entorno a estas emociones. Para facilicitar el proceso de automatización en el entorno inteligente, la plataforma estará basada en reglas semánticas ECA (Evento-Condición-Acción), permitiendo al usuario la automatización de tareas de una forma sencilla.

El sistema será implementado en un laboratorio real, donde todos los componentes y características serán probados por usuarios reales. Además, con el propósito de permitir la realización de grandes experimentos minimizando el coste, el sistema será también implementado en un escenario de simulación social. Este escenario consistirá en la simulación de una oficina inteligente, en la que las redes de sensores y las emociones del usuario serán simulados usando modelos realistas. Con este objetivo, un sistema de simulación de agentes emocionales será también desarrollado e integrado con la plataforma, proponiendo además un modelo de simulación de estrés.

Palabras clave: Simulación basada en agentes, Reconocimiento de emociones, Tecnologías semánticas, Internet de las Cosas, Python, Automatización basada en eventos

Abstract

The success of smart mobile platforms and modern wearable computers, has boosted the evolution of Internet of Things (IoT) from a far-fetched futuristic vision to a reachable reality. These devices provide valuable data coming from human beings, ambient or IoT infrastructure, enabling a new world of context aware service and applications.

However, despite the IoT emanation, and although much research has been carried out on the impact that emotions have on several domains (such as health, social relationships, working life or emergencies), there is still a lack of pervasive environments that take into account emotional behaviour. This master thesis aims to bridge this gap, and proposes the development of an IoT platform that recognises emotions, merges data and user context, and personalises the user environment according to their context.

The development of this platform is the main goal of this master thesis. The intelligent system developed must be able to capture emotions by mean of analysing data that comes from cameras and several sensors (e.g. biometric or ambient), and to automatically adapt the environment to these emotions. In order to facilitate the automation process in the intelligent environment, the platform will be based on ECA (Event-Condition-Action) rules, allowing the user to automate tasks in an easy way.

The system will be deployed into a living lab, where all components and features will be tested by real users in a real scenario. In addition, with the purpose of enabling the realization of large-scale experiments at negligible cost, the system will also be deployed into a social simulated scenario. This scenario will consist of a simulation of a smart office, where sensor networks and users' emotions will be simulated with realistic environment models. With this purpose, an emotional agent-based simulation system will also be developed and integrated with the automation platform, and a stress simulation model is proposed.

Keywords: Agent-based simulation, Emotion recognition, Semantic technologies, Internet of Things, Python, Notation3, Event-based automation

Agradecimientos

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter provides an introduction to the problem that will be approached in this project, providing a brief overview to ambient intelligence, emotion recognition and agent simulation. Given the context of the project and its main approaches, the goal of this master thesis is drafted.

The success of smart mobile platforms and modern wearable computers, has boosted the evolution of Internet of Things (IoT) from a far-fetched futuristic vision to a reachable reality. These devices provide valuable data coming from human beings, ambient or IoT infrastructure, enabling a new world of context aware services and applications. Over the last years, the number of interconnected devices on the planet has been continuously growing: in 2011, it overtooks the actual number of people; currently, there are almost 12 billion interconnected devices; and it is expected to reach 20 billion devices by 2020 [1]. The evolution of IoT, leads to new opportunities for the Information and Communication Technologies (ICT) sector, allowing new services and applications to be able to leverage the interconnection of physical and virtual realms [2].

One of these opportunities is the application of Ambient Intelligence (AmI) principles to the workplace, that turns out in the notion of smart offices. Smart offices can be defined as "workplaces that proactively, but sensibly, support people in their daily work" [3]. They should be aligned to the business objectives of the enterprise [4], and should enable a productive environment that maximizes employee satisfaction and company performance. Thus, smart offices should manage efficiently and proactively the IoT infrastructure deployed in the workplace as well as the enterprise systems. Moreover, smart offices should be able to interact with smartphones and help employees to conciliate their personal and professional communications.

A recent shift in AmI is to move from the model of full and transparent automation to smart collaboration, since autonomy often leave users feeling out of control [5]. A popular approach to interconnect and personalize both IoT as well as Internet services is the use of Event-Condition-Action (ECA) rules, also known as trigger-action rules. A number of now prominent web sites, mobile and desktop applications feature this rule-based task automation model, such as Ifttt[1] or Zapier[2]. These systems, so-called Task Automation Server (TAS) [6], are typically web platforms or smartphone applications, which provide an intuitive visual programming environment where inexperienced users seamlessly create and manage their own automations. In addition, the use of semantic vocabularies for modeling automation rules enables data interoperability and portability of automations, allowing external resource linking to the instances such as those from the Linked Open Data (LOD) cloud [7]. By mean of semantic technologies, data collected from different sources can be translated into meaningful information in order to characterize the context and the activity of the users.

Despite the IoT emanation and the successful implementation of several intelligent sys-

[1] https://ifttt.com/
[2] https://zapier.com/

tems in multiple domains, there is still a lack of pervasive environments that take into account emotional behaviour. Much research has been carried out on the impact that emotions have on health [8], emergencies [9] and working life [10]; which states the importance of recognizing and processing people emotions in intelligent environments. The application of emotion aware technologies in IoT environments entails a quantitative improvement in the users quality of life, since it allows the environment to be adaptive to these emotions and, therefore, to people needs.

Focusing on the smart office context, stress is one of the most attractive emotions to analyze, as several research has been done in order to probe its influence on the job performance. Stress is defined as the response of the body to any demand placed on an organism [11], that will be positive or negative depending on the nature of this demand [12]. The optimal work performance of an individual is attained when the negative stress (distress) reaches its minimum value and the positive stress (eustress) reaches its maximum [13]. Stress detection has been addressed from several approaches and taking into account different parameters, such as facial recognition [14], finger temperature [15], heart rate [16], skin response [17], pupil dilatation and eyetracking [18].

However, deployment of large-scale emotion aware IoT systems involves a significant cost of time and money. The logistical challenge of experimenting with thousands of small battery-powered nodes and people is the key factor that has greatly limited the development of this field. Using simulation is a good approach for overcoming these drawbacks and conducting scalability research for IoT networks. This technology allows IoT systems to be tested, verified and validated automatically by simulations before taken them to living labs or the final environment, anticipating a large number of faults before software deployment. Some of these simulation approaches are: WISEBED [19], Senslab [20] and TWIST [21]. Regarding to emotion simulation, the need of affective systems has resulted in the appearance of a large number of models related to human behaviour simulation [22]. Most of these systems (such as WASABI [23] or JBdiEmo [24]) are based on the OCC model [25], that has become the standard model for the synthesis of emotion. This model specifies how events, agents and objects are appraised according to individual's goals, standards and attitudes.

With regard to stress modelling, most works stem from the classic Yerkes-Dodson model [13]. Mayuri Duggirala et al. [26] model simulates how individuals in a team perceive, cope and respond to stress in terms of their productivity, as well as how accumulation of team backlog and virtuous cycles of productivity affect to the team. Matthew Page et al. [27] propose to encode agents initially using a String Representation and later multi-state Binary Decision Automata to choose between work on a base task, special project or rest.

1.1 Goals

The main purpose of this master thesis is the development of an IoT platform that recognizes emotions, merges data and user context, and personalizes the user environment according to their context.

The intelligent system developed must be able to capture emotions by mean of analyzing data that comes from several sources, and to automatically adapt the environment to these emotions. In order to facilitate the automation process in the intelligent environment, the platform will be based on ECA (Event-Condition-Action) rules, allowing the user to automate tasks in an easy way.

While the system will be designed with the purpose of making straightforward its integration and configuration in any place, this project focuses on its implementation in a smart office scenario. For this purpose, the system will be deployed into a living lab, where all components and features will be tested by real users in a real scenario. In addition, with the purpose of enabling the realization of large-scale experiments at negligible cost, the system will also be deployed into a social simulated scenario. This scenario will consist of a simulation of a smart office, where sensor networks and users' emotions will be simulated with realistic environment models.

With this purpose, an emotional agent-based simulation system will also be developed andintegrated with the automation platform, and a stress simulation model is proposed.

1.2 Structure

In this section we will provide a brief overview of all the chapters of this Master Thesis. It has been structured as follows:

Chapter 1 provides an introduction to the problem which will be approached in this project. It provides an overview of the importance of emotion recognition technologies in smart environments. Furthermore, a deeper description of the project is also given.

Chapter 2 an overview about emotions, describing some of the most popular approaches for modeling, the recent work on emotion recognition an relevant studies about agent-based simulation techniques. Besides, this chapter also describe some of the actual work on semantic event-driven automation.

Chapter 3 describes the design and implementation of the emotional agent-based model

of stress for workplaces. The architecture of the simulation system will be explained, as well as the most relevant parameters in which the simulation is focused on.

Chapter 4 describes the architecture of the emotion aware automation platform, including the design phase and implementation details. The modules in which the whole system is structured as well as the interaction between modules or with the users will also be explained.

Chapter 5 presents the complete case study of the developed system. The application of the system to a living lab as well as its application to a social simulated scenario will be extensively described, explaining the usage of all the tools involved and their purpose, as well as the evaluation of the system.

Chapter 6 draws conclusions from the previous results and outlines for the future work in this regard.

Finally, the appendix provide useful related information, especially covering the installation and configuration of the tools used in this thesis.

CHAPTER 2

State of the Art

This chapter offers a brief review of the main technologies that have made possible this project, as well as some of the related published works. It shows an overview about emotions, describing some of the most popular approaches for modeling, the recent work on emotion recognition an relevant studies about agent-based simulation techniques. Besides, this chapter also describes some of the actual work on semantic event-driven automation.

In order to accomplish the analysis and detection of emotions in a real environment, as well as to simulate emotional agents and having a high level of knowledge about human cognitive emotion and emotion synthesis theories results crucial. In addition, for endowing simulations with more realism, it is important to deepen works related to emotional agents.

With regards to IoT, it is imperative in the development of this project to analyze the state of art, studying the most popular approaches and technologies related to smart environments and emotion detection. Furthermore, actual research on semantic technologies will also be studied, focusing on some relevant ontologies.

2.1 Human cognitive emotion

Although providing a strict definition of emotion has been a notorious problem over the last years, according to the theory proposed by Michel Cabanac [28], emotion can be defined as any conscious experience characterized by intense mental activity and a high degree of pleasure or displeasure.

Emotions have a great influence on the performance of many of our everyday tasks, such as decision-making, communication, and behaviour [29], as they are linked with mood, temperament, personality, disposition, and motivation. They modify the brain's activity level for managing our behaviour as a response to perceived internal or external events [30].

According to some theories, emotions are syndromes of components. Klaus R. Scherer [29] proposes a model where they are composed of five crucial elements: cognitive component (an evaluation of events and objects), bodily symptoms (physiological component of emotional experience), action tendencies (motivational component for preparation and direction of motor responses), motor expression (facial and vocal expression that communicate reaction and intention of actions) and feelings (the subjective experience of an emotional state). This model provides a sequence of events that effectively describes the coordination involved during an emotional episode.

With regards to the classification of emotions, it has mainly been researched from two fundamental viewpoints. The first viewpoint is that emotions are discrete and fundamentally different constructs, while the second viewpoint asserts that emotions can be characterized on a dimensional basis in groupings. One of the most influential classification approaches was developed by Robert Plutchik. Plutchik's theory proposes the existence of eight basic emotions: joy, trust, fear, surprise, sadness, disgust, anger, and interest, that are the main emotions from which the rest derive [31]. These eight basic emotions were

grouped into four pairs of polar opposites with varying degrees of intensity (displayed as the wheel of emotions), enabling the representation of an emotional state in a four dimensional vectorial space.

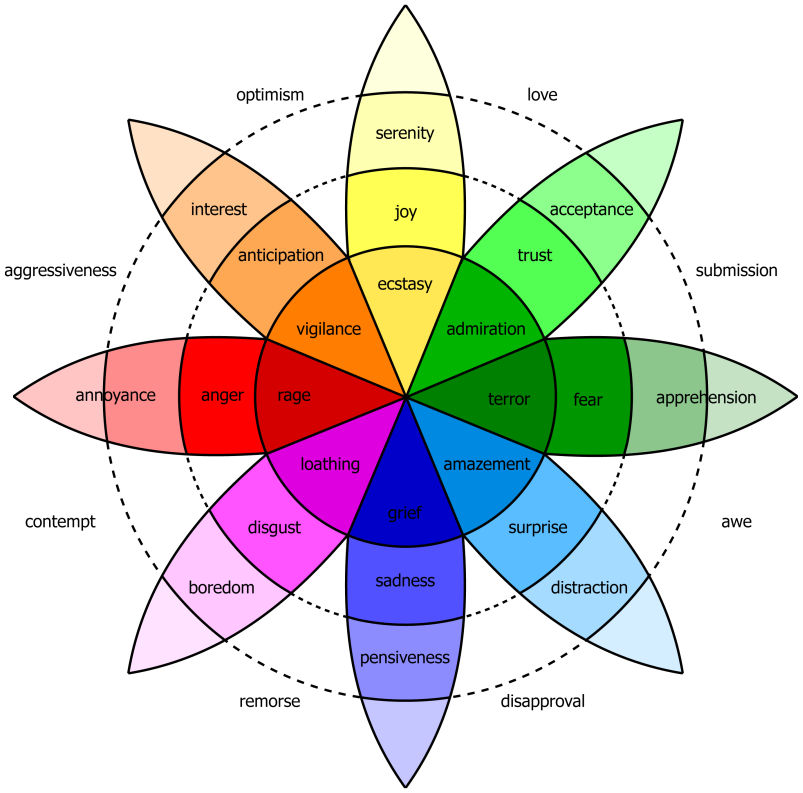

Figure 2.1: Wheel of emotions[1]

2.1.1 Emotion synthesis

Concerning to the emotion synthesis, some theories, called somatic theories, argue that bodily responses (physiological component) are necessary for emotions to occur. The first modern version of such theories is called James-Lange theory, and claims that physiological changes precede emotions, which are equivalent to our subjective experience of physiological changes, and are experienced as feelings [32]. Cannon-Bard theory argues that, even though physiological changes play a crucial role in emotions, people feel emotions first and then act according to them [33]. This theory is based on the premise that a subject is able to react to a stimulus only after perceiving the experience itself and the associated emotion. In the 1960s, Stanley Schachter suggested that physiological reactions contributed to emotional

[1]By Machine Elf 1735 - Own work, Public Domain

experience by facilitating a focused cognitive appraisal of a given physiologically arousing event and that subjective emotional experience was defined by this appraisal. Emotions are thus a result of two-stage process: general physiological arousal, and experience of emotion. In this way, Schachter, in concert with his student Jerome Singer, proposed the Singer-Schachter theory (also known as two-factor theory), that claims that subjects can have different emotional reactions despite being in the same physiological state [34].

With the two-factor theory incorporating cognition, several theories, called cognitive theories, began to argue that cognitive activity is entirely necessary for an emotion to occur. This means to imply the fact that emotions are about something or possess an intention. One of the main proponents of this view was Richard Lazarus, who stated that an emotion is a three stage disturbance that occurs in the following order: 1. cognitive appraisal - the subject assesses the event cognitively. 2. physiological change - biological changes occur as a reaction to cognitive appraisal. 3. action - the individual feels the emotion and chooses how to react [35]. Lazarus emphasized that cognitive processes control the emotions intensity and quality.

A recent hybrid of the somatic and cognitive theories of emotion is the perceptual theory, defended by Jesse Prinz [36] and James Laird [37]. The novel claim of this theory is that conceptually-based cognition is unnecessary for such meaning. Rather the bodily changes themselves perceive the meaningful content of the emotion because of being causally triggered by certain situations. In this respect, emotions are held to be analogous to faculties such as vision or touch, which provide information about the relation between the subject and the world in various ways. To sum up, the following table shows a comparison of the principal theories related to emotion synthesis, with their main ideas.

Somatic	James-Lange	Physiological changes precede emotions
	Cannon-Bard	Emotions precede physiological changes
	Two-factor	Emotions are a result of physiological arousal and experience of emotion
Cognitive	Lazarus	Cognitive activity is entirely necessary for an emotion to occur
Hybrid	Perceptual theory	Conceptually-based cognition is unnecessary

Table 2.1: Emotion theories

2.1.2 Emotion representation

Despite the fact that there are several models for emotions, ranging from the most simplistic and ancient that come from Chinese philosophers to the most modern theories that refine and expand older models, none of them has achieved to be universally accepted [38].

One of the most notable emotion annotation and representation languages is EmotionML[2], born from the efforts made for Emotion Annotation and Representation Language (EARL) [39] by Human Machine Interaction Network on Emotion (HUMAINE). EARL originally included 48 emotions divided into 10 different categories. EmotionML offers twelve vocabularies (set of possible values for any given attribute of the emotion) for categories, appraisals, dimensions and action tendencies.

In the field of Semantic Technologies, Grassi introduced Human Emotion Ontology (HEO), an ontology for human emotions meant for annotating emotions in multimedia data. Another work worth mentioning is Emotion Ontology (EMO), an ontology that tries to reconcile the discrepancies in affective phenomena terminology [40]. It is, however, too general to be used in the context of emotion analysis: it provides a qualitative notion of emotions, when a quantitative one would be needed. In this project, the emotions will be modeled using Onyx[3], an ontology designed to annotate and describe the emotions expressed by user-generated content on the web or in particular Information Systems.

2.1.2.1 Onyx

Onyx is a vocabulary that models emotions and the emotion analysis process itself [41], that can be used for representing the results of an emotion analysis service. An advantage of the model of emotions in Onyx, is that is generic model, enabling the separation of representation and psychological models.

The goals of the Onyx ontology to achieve as a data schema are: to enable the publication of raw data about emotions in user-generated content; deliver schema that will allow to compare emotions coming from different systems; interconnect emotions by linking them to contextual information expressed with concepts from other popular ontologies or specialised domain ontologies.

At its core, the ontology has three main classes: *Emotion*, *EmotionAnalysis* and *EmotionSet*. In a standard emotion analysis, these three classes are related as follows: an

[2]https://www.w3.org/TR/emotionml/
[3]Onyx ontology: http://www.gsi.dit.upm.es/ontologies/onyx/

EmotionAnalysis is run on a source, the result is represented as one or more *EmotionSet* instances that contain one or more *Emotion* instances each.

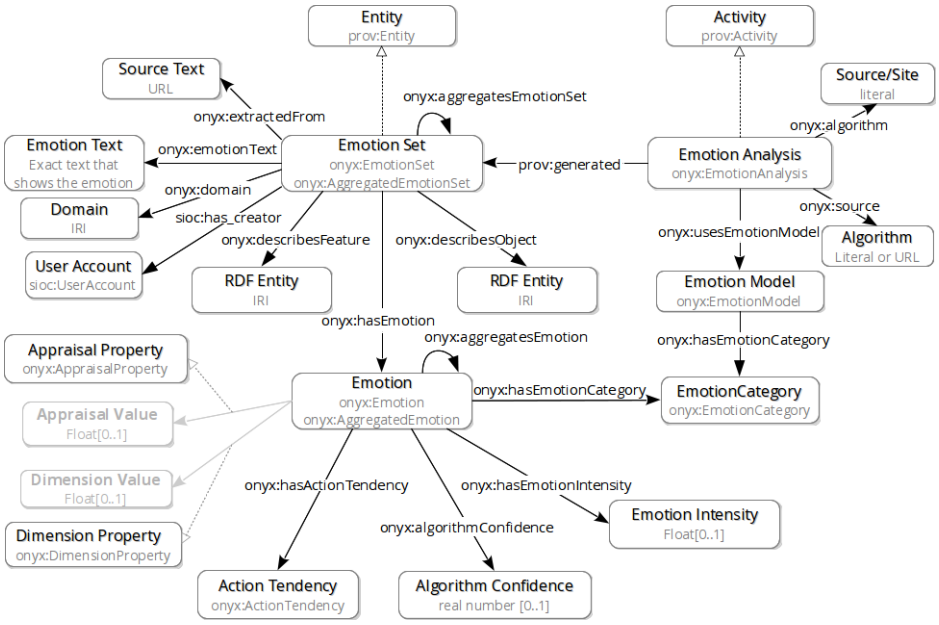

Figure 2.2: Onyx ontology overview

Onyx's Emotion model includes: *EmotionCategory* which is a specific category of emotion (e.g. "sadness", although more than one could be specified), linked through the *hasEmotionCategory* property; the emotion intensity via *hasEmotionIntensity*; action tendencies related to this emotion, or actions that are triggered by the emotion; appraisals and dimensions. Lastly, specific appraisals, dimensions and action tendencies can be defined by sub-classing *Appraisal*, *Dimension* and *ActionTendency*, whose value should be a float number.

Having a formal representation of the categories and dimensions proves very useful when dealing with heterogeneous datasets in emotion analysis. In addition to being necessary to interpret the results, this information can be used to filter out results and for automation.

2.1.3 Agent simulation models

Agent-based modeling refers to the simulation of the behavior and interaction of autonomous entities, or agents, over time. Agents, which represent individual organisms, humans, entire organizations, or abstract entities, are objects that have rules and states, and act accordingly with each step of the simulation [42].

Although several emotion models have been designed over the last years, the OCC

model has become the standard model for the synthesis of emotion. This model specifies how events, agents and objects are appraised according to individual's goals, standards and attitudes.

2.1.3.1 Ortony Clore & Collins (OCC)

This model proposes a cognitive appraisal theory that is composed of three branches, namely emotions concerning consequences of events, actions of agents, and aspects of objects. The intensity of a particular emotion depends on which of these objects, agents and events are present in the environment [25]. This intensity is determined by three variables: desirability of an event (the congruence of its consequences with the individual's goals), praise of an action (its conformity to norms and standards) and the attraction of an object (the correspondence of its aspects with the individual's likings). Additionally, some branches combine to form a group of compound emotions, namely emotions concerning consequences of events caused by actions of agents.

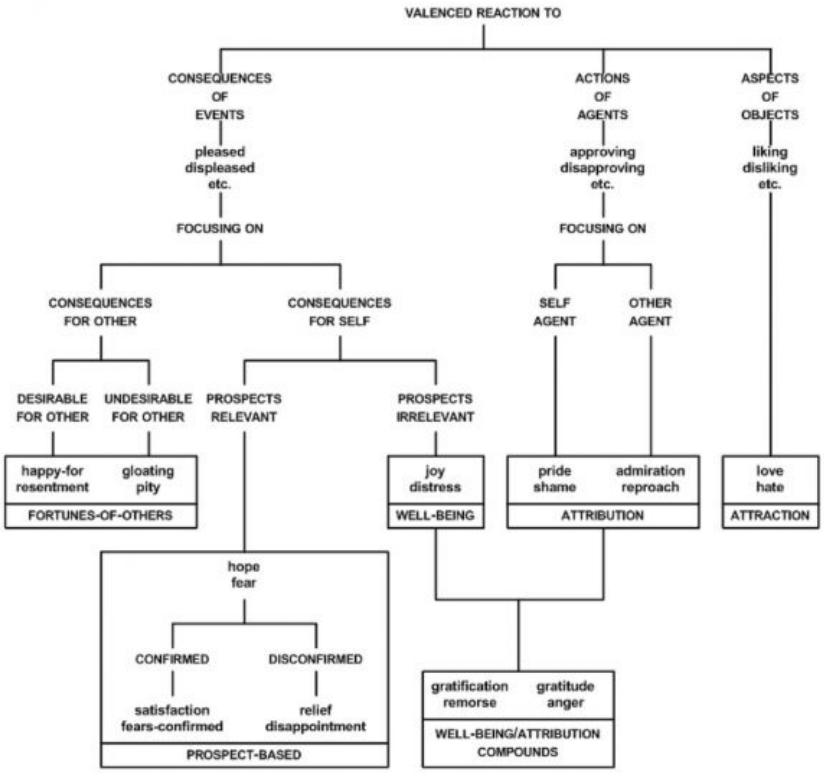

Figure 2.3: Structure of emotions of the OCC model[4]

The authors also define a set of intensity variables: sense of reality, proximity, unexpectedness and arousal are the global variables that influence all three emotion categories. There is also a threshold value for each emotion, below which an emotion is not subjectively felt, but once an emotion is felt, it influences the subject. In this way, OCC model argues that behavior is a response to an emotional state in conjunction with a particular initiating event

The vast majority of current emotion simulation models are based on this model, and is quite popular among computer scientists who are developing systems that incorporate emotions in artificial characters.

2.1.3.2 Connectionist model (Soar)

In this model, emotions are regarded as subconscious signals and evaluations that inform, modify, and receive feedback from a variety of sources including higher cognitive processes and the sensoriomotor system [43]. The emotional system is mainly composed of five components: clarity, confusion, pleasure, pain and arousal. The former four work together in order to detect events of importance to an agent, while the arousal system functions as a kind of interface between the emotional and higher cognitive systems. Instead of creating separate systems for each emotion, this model assumes that humans attach emotional labels to various configurations of these factors.

Clarity and confusion represent the agent's current knowledge and abilities, and give it an assessment about its faculty to deal with the current situation. Pleasure and pain interpret the level to which a stimulus represents a threat or enhancement for the agent. The Soar model emphasizes the effects of emotions on cognition and decision making.

2.1.3.3 BDI model

The BDI model, based Michael Bratman's theory of human practical reasoning [44], is characterized by the implementation of an agent's beliefs (the informational state of the agent), desires (the motivational state of the agent) and intentions (the deliberative state of the agent). The model also includes events, which are triggers that may update beliefs, desires or intentions. In essence, it provides a mechanism for separating the activity of selecting a plan from the execution of currently active plans. Consequently, BDI agents are able to balance the time spent on deliberating about plans and executing those plans.

[4]OCC structure, taken from The Cognitive Structure of Emotion [25]

This model has been extended in multiple occasions. One of the most popular extensions, named EBDI, proposes an extension in which agent behavior is guided not only by beliefs, desires and intentions, but also by emotions in the processes of reasoning and decision-making [45].

2.1.3.4 Further models

Actually a great number of emotion simulation models exists in several scopes. Deep [46]is an emotion dynamics model that stems from OCC model, and takes into account the personality and social relations of the agents. The model, specifically designed to be used in the video game industry to improve the credibility of Non-Player Characters, focuses on the influence of personality in triggering emotions and the influence of emotions in inter-character social interactions.

PETEEI [47]model emphasize on incorporating various learning mechanisms so that it can produce emotions according to its own experience, and is also able to recognize and react to various moods and emotional states. This model uses Q-learning with the reward treated as a probability distribution to learn about events.

2.1.4 Simulation tools

In the last few years, several tools and frameworks have been developed and are currently in wide use for agent-based modeling. There are frameworks designed for both general purpose and particular domain, and each of them with different features and characteristics as programming languages, license or learning curve.

2.1.4.1 NetLogo

One of these tools is NetLogo [48], a multi-agent programming language and modeling environment for simulating natural and social phenomena. It is written in Java, and is particularly well suited for modeling complex systems evolving over time.

It comes with an extensive models library including models in a variety of domains, such as economics, biology, physics, chemistry, psychology, system dynamics. NetLogo enables the exploration through its graphic interface, modifying switches, sliders, choosers, inputs, and other interface elements. Beyond exploration, NetLogo allows authoring of new models and modification of existing models.

2.1.4.2 Mason

MASON [49]is a single-process discrete-event simulation core and visualization library written in Java, designed to be flexible enough to be used for a wide range of simple simulations. The main design goal was to build a fast, orthogonal, minimal model library to which an experienced Java programmer could easily add features, rather than one with many domain-specific, intertwined features which are difficult to remove or modify. In addition, it provides visualization and GUI facilities that are useful for various simulation tasks.

2.1.4.3 Mesa

However, taking into account the increasingly popularity gained by Python language for scientific computing [50], results interesting the use of a framework written in this language. Python is supported by a mature and growing ecosystem of tools for analysis and modeling, and is widely considered a more natural, easy-to-use language than Java. Mesa is a new open-source Python package that allows users to quickly create agent-based models using built-in core components or customized implementations; visualize them using a browser-based interface; and analyze their results using Python's data analysis tools [51].

Mesa modules are divided into three overall categories: modeling, analysis and visualization. The modeling components are the core of what's needed to build a model: Model class (that stores model-level parameters and serves as a container for the rest of the components), Agent classes (that describe the model agents), and scheduler (that controls the agent activation regime, and handles time in the model).

In addition, Data Collector class is used to record data from each model run, as can store and export data from most models. The data collector stores three categories of data: model level variables, agent-level variables, and tables which are a catch-all for everything else. Internally, the data collector stores all variables and tables in Python's standard dictionaries and lists. Batch Runner class enables the automation of multiple runs and parameter sweeps.

2.1.4.4 SOBA

SOBA [52] is a Simulator of Occupancy Based on Agent useful for developing studies and research based on social simulation, mainly in buildings. This tool provides a module for modelling occupancy in buildings, that enables the definition of several types of occupants with different behaviors, a physical space and the interconnection between agents.

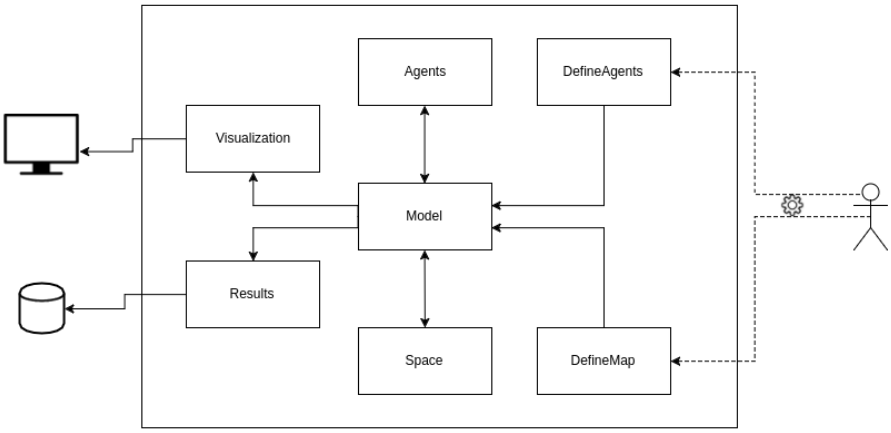

Figure 2.4: SOBA architecture

Figure 2.4 shows an overview of SOBA architecture. This project will use the Space module of SOBA in order to define and modelling the building where the agents will be located.

2.1.4.5 RAMEN

RAMEN [53]is a tool based on Blueprint3D [5], that enables the visualization of the simulation (with all its agents, devices and places) in a 3D environment. The integration of the simulation system with RAMEN makes simpler and faster to find solutions for possible problems or to reach conclusions from the simulation.

This tool will create a 3D model of a specific building interior with its different agents and devices, enabling the opportunity to see how different aspects vary depending on human behaviors, as the temperature variation of a room or the interactions between humans and sensors.

RAMEN is composed of three main modules: *Batch module*, *Model generator* and *Visualization module*:

Batch module: this module is in charge of collecting the data given by the user, such as the floor plan, the rooms assignation and the movement of the agents, and converting it into the objects of the visualization module.

Model generator: thanks to the data collected by the batch module, it is able to generate the scene in which the different elements are represented. The main elements are the

[5]Blueprint3D: https://github.com/furnishup/blueprint3d

floor plan, the agents, the items and the camera. All of them make use of the renderer which provides the data to represent the visualization.

Visualization module: this module is in charge of representing and managing the final result of the visualization on the browser.

The most relevant module for this project is the batch module, as it is responsible for collecting the data from the simulation system that will make up the visualization. This module receives several JSON files in order to build and represent the place along with the agents and devices that are going to step in the simulation. The files that the user has to proportionate are the following ones:

- Floorplan/Items: defines all the floor plan and the items in it. In addition, the floor plan is compounded by all the corners of the map and the walls between them.

- Rooms: defines the names of the rooms, in order to identify all the rooms by the name that the user has chosen. It has to be declared the name and the coordinates of the center of the room.

- Movement: declares all the actions that occur during the simulation. It is divided in steps and in each one, all the actions that happen in that moment are defined. The steps are a way of measuring time and everyone has the same duration, which is defined by the simulator.

2.1.5 Emotion detection

Over the last years, emotion detection represents a significant challenge that is gaining the attention of a greate number of researchers. The main goal is the use of different inputs for carrying out the detection and identification of the emotional state of a subject. Emotion recognition opens endless possibilities as it has wide applications on several fields such as health, emergencies, working life, or commercial sector.

The traditional approach of detect emotions through questionnaires answered by the participants does not result very efficient method. That is the reason for focusing on automatic emotion detection using other physiological responses, such as heart rate, skin conductance, pupil dilation, and by mean of several techniques such as facial recognition, speech analyzing, or Electroencephalographic (EEG).

2.1.5.1 Face recognition

Algorithms to predict emotions based on facial expressions are mature and considered accurate. Currently there are two main techniques to realize facial expression recognition depending on its way of extracting feature data: appearance-based features, or geometry-based features. Geometric based technique finds a set of representative features of geometric form to represent an object by collecting geometric features from images. In this technique, the shape of the face and its components is described and emotional data is extracted from motion of facial muscles. Otherwise, appearance based extraction describes the texture of the face caused by expression, and extracts emotional data from skin changes.

As commented above, in geometric feature-based approaches the primary step is to localize and track a set of facial points. For this purpose, most of approaches use the active appearance model (AAM) or its variations. There is a big variety of ways in the current literature for extracting the shape and movement of facial features from the facial points. Choi et al. [54] proposed a system able to recognize the facial expressions of a person in five classes (neutral, happy, sad, angry, and surprise) by mean of the use of AAM with efficient second order minimization, and a multilayer perceptron. Sebe et al. [55] proposed a method where the facial feature points were manually located, and then tracked using piecewise Bezier volume deformation (PBVD). This tracker uses a model-based approach where an explicit 3D wireframe model of the face is constructed. Sung and Kim [56] proposed a model that improves the fitting and tracking of standard AAMs, called Stereo AAM (STAAM). STAAM uses a geometric relationship between two tightly coupled views multiple cameras to model the 3D shape and rigid motion parameters, increasing the accuracy and speed of fitting. A layered generalized discriminant analysis (GDA) classifier is then used for recognizing facial expression through the combination of 3D shape and registered 2D appearance.

With regards to appearance based extraction, there are several techniques that have been successfully employed for emotion recognition such as local binary pattern (LBP) operator, linear discriminant analysis (LDA), histogram of orientation gradients (HOG), non-negative matrix factorization (NMF) based texture feature, principle component analysis (PCA), etc. LBP operator is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considering the result as a binary number. It was used along with support vector machine (SVM) by Sahn et al. [57], obtaining a great facial expression recognition accuracy. LDA is a popular face recognition technique that aims to find the most discriminative features maximizing the ratio of determinant of between-class variations to within-class variations. HOG lies in dividing the image into

small connected regions called cells, and for the pixels within each cell, compile a histogram of gradient directions. PCA method is based on transforming the face images into a small set of characteristics feature images, called eigenfaces, which are the principal components of the initial training set of the face images.

There are also hybrid systems that have the advantages of both geometric-based and appearance-based features. Lyons and Akamatsu [58] introduced a system for coding facial expressions using a multi-orientation, multi-resolution set of Gabor filters at some fixed geometric positions of the facial landmarks, obtaining a similarity space that was compared with one derived from semantic ratings of the images by human observers.

2.1.5.2 Speech analyzing

When analyzing the speech for detecting emotions, most of approaches use pitch, speech rate and intensity of speech. Pitch is a basic element of speech signal in the frequency domain, speech rate is computed through number of samples, and the intensity of the speech signal is computed through the strength of the input speech signal.

However, to take into account further parameters such as Mel Frequency Cepstral Coefficients (MFCC's), Perceptual Linear Predictive Cepstrum (PLPC) , Mel Frequency Perceptual Linear Predictive Cepstrum (MFPLPC) and Linear Predictive Coefficients (LPC) may improve significantly the accuracy of the analysis as demonstrated by S. Lalitha and S. Tripathi [59], obtaining an accuracy around 85% combining them with pitch, speech rate and intensity.

With regards to classification, the most common classification methods used in emotion recognition are based on Hidden Markov Models (HMM) [60], artificial neural networks (ANN) [61], linear discrimination analysis (LDA) [62], and support vector machine (SVM) [63].

In addition, further approach to emotion recognition by mean of speech analyzing is the transcription of the speech to text, and the subsequent emotion extraction of the text. Several tools have been developed in order to carry out the emotion extraction of text, but this project focuses on Senpy.

Senpy is an open source software and uses a data model for analysis of feelings and emotions [64]. It is based on NIF, Marl and Onyx vocabularies, and aims to facilitate the adoption of the proposed data model to analyze feelings and emotions, in order to improve the interoperability between services of different providers.

The framework consists of two main modules: Senpy core, which is the building block of the service, and Senpy plugins, which consists of several NLP algorithms. Some of the plugins defined in Senpy platform are:

- *emoTextAnew*: it extracts the VAD (Valence - Arousal - Dominance) of the sentence provided by matching words from the ANEW dictionary [65]. Once the VAD triple is obtained, it can be compared with the nearest emotion.
- *emoTextWordnetAffect*: it is based on the hierarchy of WordnetAffect [6] to obtain the emotion of the sentence provided.
- *sentiText*: it refers to a software developed during TASS 2015 competition [7] and has been adapted for multilingual purpose for English and Spanish.
- *affect*: created to perform a sentiments and emotion analysis simultaneously.

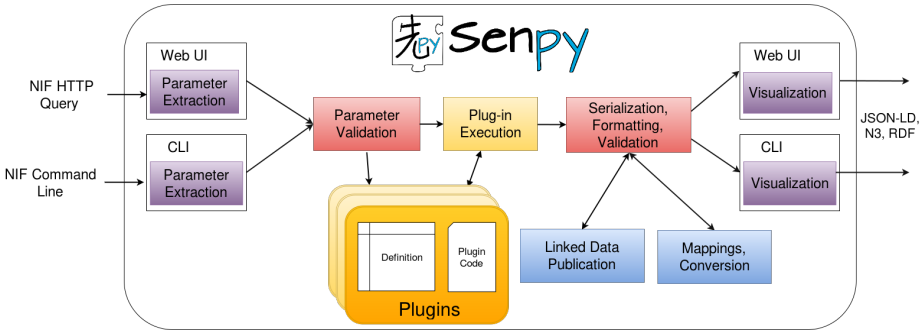

Figure 2.5: Senpy architecture

Figure 2.5 shows a simplified version of the processes involved in the analysis with the Senpy framework. The tool extracts the parameter from a NIF HTTP query and executes the code of the plugin selected with the inputted parameters previously validated. Then, use models to output a linked data publication in the desired format.

2.1.5.3 Physiological signals

Although some research has been conducted in this field, mostly in the study of EEG signals, it represents a less explored area than speech and face recognition. Some of the most used features are the following:

[6] http://wndomains.fbk.eu/wnaffect.html
[7] http://www.sepln.org/workshops/tass/2015/tass2015.php

Heart Rate (HR) The increase and decrease of HR is associated to many emotions, thus the average HR computed over the whole duration of an emotional stimulus is a feature that can be used for emotion assessment.

EEG Most of the current approaches of emotion detection by mean of EEG signal analysis obtain results by comparing the energy in different frequency bands, as the power spectral densities of EEG signals in different bands are correlated with emotions [66]. In addition, the are also specific patterns of synchronization between brain areas during emotional processes [67].

Galvanic Skin Response (GSR) The correlation between mean skin resistance and the arousal of a stimulus has been studied and demonstrated [68], so GSR has been widely used for emotion assessment from peripheral physiological signals. When an emotion arises, there is a decrease in the GSR signal.

2.2 Semantic Event-driven Automation

Rules are representations of knowledge with conditions in some domains of logic, such as first-order logic. A rule is basically defined in form of If-then clauses containing logical functions and operations, and can be expressed in rule languages or formats. Generally, a rule consists of antecedence and consequence containing clauses, and logical quantifiers used to quantify possible domains of variables. The antecedence contains conditions combined using logical operators, while the consequence part contains conclusions (or actions). If all the conditions are matched, the conclusions are operated. A clause is in the form of subject-relation-object, where subject and object can be variables, individuals, literal values or other data structures [69].

A number of now prominent web sites, mobile and desktop applications feature rule-based task automation. Typically, these services provide users the ability to define which action should be executed when some event is triggered. Some examples of this simple task automation could be “When I am stressed at work, play relaxing music” or “When I feel sad, adjust the room light color”. We call them Task Automation Services (TASs).

2.2.1 Task Automation Services

These services provide a visual programming environment, where non-technical users can seamlessly create, manage or even share their own personal automations. The automation in these services takes the form of Event-Condition-Action rules that execute an action upon a

certain triggering event i.e. “when triggering-event then do action”. In the former examples, the detection of stress at work and the detection of sad would be the triggers, whereas playing relaxing music and adjusting the room light color are the respective actions. The use of TASs has grown substantially, own to three main factors: usability (TASs provide a simple-yet-powerful intuitive interface for programming task automations), customisability (TASs allow their users to program the automations they need) and integration with existing Internet services (users can automate tasks that access the Internet services they already use and are familiar with) [6].

Since multiple TASs exist, the execution strategy defined as rules is completely independent of the actual reasoning platform. Moreover, as both decision trees and rule-based reasoning follow the same kind of thinking (i.e., a sequence of 'when this then that' steps), writing rules to specify decision trees comes very natural. The used engine should be very expressive, to maximize configurability, and there's a language that stands out from the rest in this point: Notation3.

2.2.2 Notation3

Notation3 (N3) [70] is a Semantic Web logic. Being a superset of Turtle – a serialization format of RDF data – it is capable of describing everything using triples, but also capable of describing rules to be executed on those triples. The aims of N3 are: to optimize expression of data and logic in the same language; to allow rules to be integrated smoothly with RDF; to allow quoting so that statements about statements can be made and to be as readable, natural, and symmetrical as possible.

The language achieves these goals with the following features:

- URI abbreviation using prefixes which are bound to a namespace (using @prefix) a bit like in XML.
- Formulae allowing N3 graphs to be quoted within N3 graphs using { and }.
- Variables and quantification to allow rules to be expressed.
- A simple and consistent grammar.

N3 differentiates itself from other rule languages because of its expressiveness. For example, in N3 it is possible to create rules in the consequence, and to use built-ins. The N3 logic has monotonicity of entailment, which means that the hypotheses of any derived fact may be freely extended with additional assumptions, which is an important property when

reasoning about a changing knowledge base. The expressiveness of rule-based reasoning engines depends on which logic the underlying programming language supports, and on the inherent expressiveness of the rule language. Some engines like RDFox, FuXi7 or EYE support N3 syntax, but EYE reasoner is the only which support all N3's expressitivity.

2.2.3 EYE

The EYE (Euler YAP Engine) [71] reasoner is a high-performance reasoning engine that uses an optimized resolution principle, supporting forward and backward reasoning and Euler path detection to avoid loops in an inference graph. It is written in Prolog and supports, among others, all built-in predicates defined in the Prolog ISO standard. Backward reasoning with new variables in the head of a rule and list predicates are a useful plus when dealing with OWL[8] ontologies, so is more expressive than RDFox or FuXi, whilst being more performant than other N3 reasoners.

Internally, EYE translates the supported rule language, Notation3, to Prolog Coherent Logic intermediate code and runs it on YAP (Yet Another Prolog) engine, a high-performance Prolog compiler for demand-driven indexing. The inference engine supports monotonic abduction-deduction-induction reasoning cycle. EYE can be configured with many options of reasoning, such as not proving false model, output filtering, and can also provide useful information of reasoning, for example, proof explanation, debugging logs, and warning logs. The inference engine can be added new features by using user-defined plugins.

2.2.4 EWE Ontology

EWE is a standardized data schema (also referred as "ontology" or "vocabulary") designed to describe elements within TASs enabling rule interoperability. Referring to the EWE definition [72], the goals of the EWE ontology to achieve are: to enable to publish raw data from TASs (Rules and Channels) online and in compliance with current and future Internet trends; enable rule interoperability; and provide a base vocabulary for building domain specific vocabularies.

The ontology has four main classes: Channel, Event, Action and Rule.

Channel. It defines individuals that either generate Events, provide Actions or both.

[8]Web Ontology Language: the ontology description standard as defined by the World Wide Web Consortium (W3C)

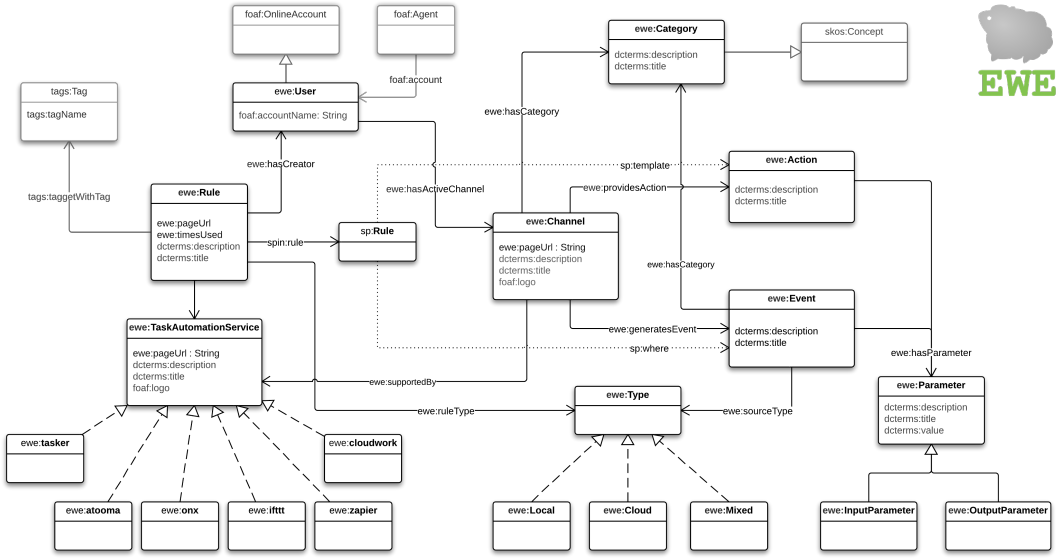

Figure 2.6: EWE Class Diagram

Sensors and actuators are also described as channels, thus they produce events or provide actions.

Event. This class defines a particular occurrence of a process. Events are instantaneous: they have no duration over time. Event individuals are generated by a certain Channel, and they are triggered by the occurrence of the process which defines them. Events usually provide further details that can be used within Rules to customize Actions: they are modelled as output parameters. Events also let users describe under which conditions should they be triggered. These are the configuration parameters, modelled as input parameters. Event definitions are not bound to certain Channels since different services may generate the same events [73].

Action. This class defines an operation or process provided by a Channel. Actions provide effects whose nature depend on itself. These include producing logs, modifying states on a server or even switching on a light in a physical location. By means of input parameters actions can be configured to react according to the data collected from an Event. These data are the output parameters.

Rule. This class defines an "Event-Condition-Action" (ECA) rule. This rule is triggered, and means the execution of an Action. Rules define particular inter-connections between instances of the Event and Action classes; those include the configuration parameters set for both of them: output from Events to input of Actions.

CHAPTER 3

Emotional Agent-based Simulation Model for Stress

This chapter describes the design and implementation of the emotional agent-based model of stress for workplaces. First, the model design will be described, explaining the different components that contribute to workers stress and how this contribution is calculated. Then, the architecture of the simulation system will be explained, focusing the most relevant components that compose it and the most relevant parameters in which the simulation is based on.

Agent-based modelling has recently incorporated key emotional and social dimensions of behavior in order to achieve a better accuracy and realism in the simulations. However, because of the complexity of demeanour in different contexts, modelling various dimensions of behavior, relations among those dimensions and impact of these dimensions on outcomes remains an open research challenge [74].

Focusing on the workplace context, one of the most relevant emotions to analyze is stress, as its negative effects on an employee have a substantial impact on productivity loss [75] and the significant correlation between mental fatigue and impairment of physical performance in humans [76]. Stress is defined as the physiological or psychological response of the body to any external, chemical, biological or environmental demand placed on an organism that prompts a response [11]. These demands are called stressors, and the response to them may be positive or negative depending on their nature.

With regards to the impact of stress on work performance, psychologists Robert Yerkes and John Dodson proposed the Yerkes-Dodson law, which is an empirical relationship between stress and performance in an attempt to obtain optimal performance from an individual [13].

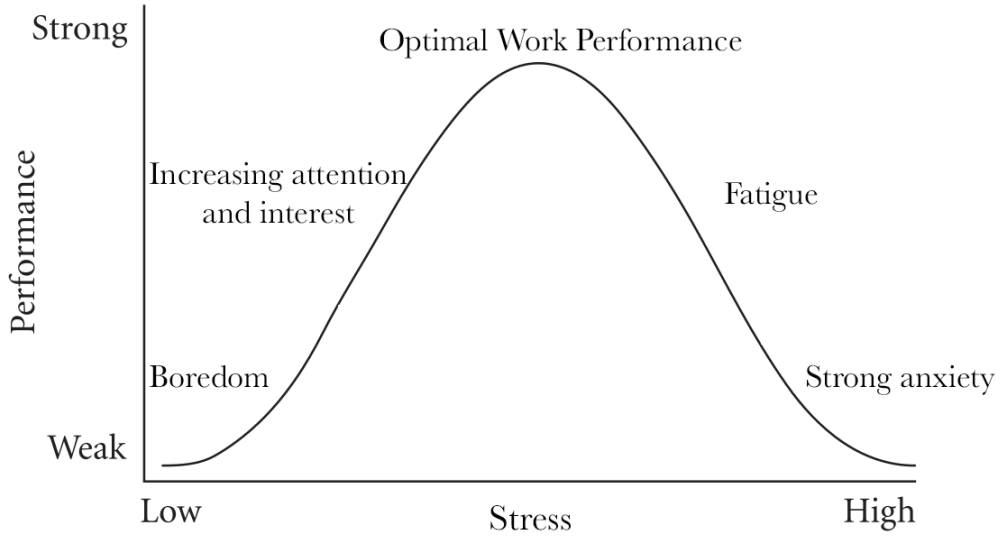

Figure 3.1: The Yerkes-Dodson pressure performance curve

Yerkes-Dodson law argues that individual performance increases with stress until the maximum of the individual is reached, and decreases if this maximum is exceeded. This law is often depicted as a normally distributed curve on a graph as seen in Figure 3.1.

The approach to model composition that is proposed in this master thesis will be described in the following sections. The model uses behavioral relations from several sources and different studies. These studies have identified several sources of work stress, such as physiological sources, as illness; ambient conditions, such as temperature, luminosity or humidity; and job-related, such as long working hours, work load, time pressure, email reception, role ambiguity, etc.

3.1 Model design

The developed stress model is based on an own interpretation of Silverman's stress model [77], which argues that stress consists of three main components: *Event Stress (ES)*, *Time Pressure (TP)*, and *Effective Fatigue (EF)*. In the context of a smart office, *Event Stress* refers to the arrival of high volume of work, *Time Pressure* refers to the relation between the time needed to complete the pending tasks in an agent's task queue and the remaining work time, and *Effective Fatigue* refers to the accumulated tiredness that results of the execution of work related tasks and the impact of external conditions. In the current work have been assumed that all of these components contribute equally to the total stress, that is calculated as a weighted sum of each one of its components.

$$S = \frac{ES + TP + EF}{3} \tag{3.1}$$

All the elements that compound stress can vary between 0 and 1.0, with 0 being overconfident and unstressed, 0.5 neutral and 1.0 representing the state of maximum stress. *Event Stress* is calculated as a relation between the number of tasks arrival in a day and the mean task arrival which the agent is used to, as shown in Equation 3.2, where NTA represents the Number of Task Arrived and MTA the Mean Task Arrival.

$$ES = \frac{\frac{NTA}{2}}{MTA} \tag{3.2}$$

Thus, if an agent whose average number of daily tasks is 20, one day receives 30, the contribution of ES will be 0.75, while if other day receives 10, the contribution will be 0.25. *Time Pressure* derives from the relation between the required time to perform accurately a task and the available real time for the task, as shown in Equation 3.3, where T_i represents the ideal time required to perform a task and T_a represents the available time.

$$TP = \frac{T_i}{(T_i + T_a + 60)} \tag{3.3}$$

Thus, if an agent has pending tasks whose ideal performing time is 30 minutes, and only makes 20 minutes available, the stress effect of TP will be 0.27. Lastly, *Effective Fatigue* consists of the contributions of several parameters called Performance Moderator Functions (PMF), that may be positive or negative stressors. Negative stressors (those which increase effective fatigue) include ambient temperature and humidity, emails received, sleep deprivation, noise, etc; while positive stressors (those which decrease effective fatigue) may be relaxing music, video, rest time, etc. Specifically, in this project the PMFs included will be overtime hours, temperature, humidity, noise, rest time, music, emails received and luminosity. EF is calculated as a normalized sum of all PMFs as shown in Equation 3.4, where $WPFM_i$ represents the weight of the PMF, FT_i represents the fatigue tolerance of the agent to the PMF ($0 \leq FT_i \leq 1$), and N represents the number of PMFs.

$$EF = \frac{1}{N}\sum_{i=1}^{N}(\overbrace{\frac{WPMF_i}{WPMF_i + FT_i}}^{\text{negative stressors}} - \underbrace{\frac{WPMF_i}{10}}_{\text{positive stressors}}) \tag{3.4}$$

In order to include the most relevant PMFs for the smart office context, the current literature about stress sources has been exhaustively analyzed. The drawn conclusions from the most relevant factors in the design of the simulation model are explained below:

Temperature and humidity: an increase or a decrease at ambient temperature can result in stress for the worker, causing a low performance capacity. The most widely used heat stress index for workplaces is wet-bulb globe temperature (WBGT) [78]. WBGT is a type of apparent temperature used to estimate the effect of temperature and humidity on humans, whose simplified form (published by the Australian Bureau of Meteorology[1]) can be calculated from ambient temperature (T_a) and water vapour pressure (V_p), as shown in Equation 3.5.

$$WBGT = 0.567 \cdot T_a + 0.393 \cdot V_p + 3.94 \tag{3.5}$$

V_p can be calculated as a function of the temperature and relative humidity (RH), as shown in Equation 3.6. Relative humidity refers to the amount of the moisture in the air, compared to the potential saturation level.

$$V_p = 6.105 \cdot \frac{RH}{100} e^{\frac{17.27 \cdot T_a}{237.7 + T_a}} \tag{3.6}$$

[1]Australian Bureau of Meteorology: www.bom.gov.au

Studies have demonstrated that the optimal level for the temperature factor is around 24°C with relative humidity of 40% [79], resulting in a WBGT of 22,23°C. In addition, it has been estimated that stress level increases between 4 and 9 percent per degree on days when WBGT is above 27°C [80]. If the WBGT level is quite high or low, it will perceived as a negative stressor and will increase effective fatigue, however, if the WBGT level is in the ideal limits, between 20°C and 25°C, it will be perceived as a positive stressor, decreasing effective fatigue.

Overtime hours: one way of coping with a demanding work situation is to work longer hours. However, the relationship between fatigue and number of work hours has been widely documented in past research, demonstrating that overtime and longer working hours result in higher levels of fatigue, errors and lower productivity. Singh et al. [81] uses a relation where fatigue level increases around 1% for each overtime working hour, so it will be considered a negative stressor.

Rest time: rest breaks have been proposed as a mean of reducing discomfort, indicating that short breaks may be beneficial for worker productivity and well-being at work. In this way, rest time will be considered a positive stressor that acts as the opposite to work overtime hours.

Noise and music: numerous research studies have confirmed noise as a primary cause of reduction in workers productivity, as it is regarded as a source of distraction, frustration and stress. In addition, it can contribute to stress and illness which, in turn, may produce absenteeism and turnover of staff [82]. A study published in the British Journal of Psychology, demonstrated that workers exposed to noise reduced by approximately 67% their accuracy [83]. Ideal noise level at work must be between 48 and 52 dBA, and a level greater than 65 dBA involves an important reduction of productivity [84], so high noise levels will be considered as a negative stressor. On the other side, listening to relaxing music prior to a stress task differentially affects biological stress response domains, and helps to recover from a stressor more efficiently [85]. In this way, music will be considered a positive stressor.

Email reception: people who daily receive a great number of emails perceive email as a great source of stress, experiencing lower job satisfaction [86] and reporting greater work overload, so it will be considered as a negative stressor. Kushlev et al. demonstrated that people experienced reduced stress when the number of times they checked their email was limited [87]. In addition, email reception affects to workers productivity as it involves an interruption of the tasks in which they were working, taking an average of 23 minutes to get back to the task [88].

Luminosity: several studies have revealed that bad luminosity may cause visual discomfort, fatigue, and mood changes, so luminosity is an important fact that can affect people emotions. Improving the visual performance and comfort will have a positive impact on human performance and, therefore, on workers productivity. Rasdan et al. stated the importance that illuminance has on productivity, and argued that the ideal illumination in the workplace is around 600 lux [79]. In this way, liminosity will be considered as a positive stressor if the light is appropiate (between 450 and 750 lux).

With the purpose of designing the model as reliable as possible, the contributions of the explained features to workers stress have been estimated following the explained researches, as shown in Table 3.1.

Feature	**Contribution**	**Description**
Temperature and humidity	If WBGT $\leq$ 20 ^{o}C or WBGT $\geq$ 25 ^{o}C: $WPMF_t = 0.04 \cdot \lvert WBGT - 22 \rvert$	An increase in the difference between the current WBGT and the ideal WBGT (22 ^{o}C) will increase the effective fatigue (E. Somanathan et al.)
Overtime hours	$WPMF_o = 0.021 \cdot O_h$	Working overtime hours (O_h) will increase the effective fatigue (M. Singh et al.)
Rest time	$WPMF_r = 0.016 \cdot R_t$	Taking a break (R_t) will decrease effective fatigue
Noise	If $N_L \geq 65$ dB: $WPMF_n = 0.03 \cdot (N_L - 65)$	If the noise level (N_L) increases over a certain limit (65 dB), the effective fatigue will also increase (L. E. Maxwell)
Email reception	$WPMF_e = 0.0029 \cdot E_r$	The effective fatigue will increase with the number of emails received (Er) (K. Kushlev et al.)
Luminosity	If $L \leq 450$ lux or $L \geq 750$ lux: $WPMF_l = 0.000153 \cdot \lvert L - 600 \rvert$	If the difference between the current luminosity (L) and the ideal luminosity (600 lux) increases, the effective fatigue will also increase (A. Rasdan et al.)

Table 3.1: Performance Moderator Functions contribution to effective fatigue

The above parameters influence stress, as they influence one of its components (effective fatigue). Once the three components of stress are calculated, the stress can be determined. However, in addition to the calculation of the stress level of the worker, the rate at which this stress impacts worker's productivity results very interesting, as productivity will be usually the feature to maximize in a company. The integration of the Yerkes-Dodson "inverted-U" model of stimuli and performance enables the description of the relation between stress and performance. The inverted-U model shows that in order to achieve the maximum performance it is necessary to have an optimal arousal level. If the arousal level overcomes that optimal level or if it is not high enough, performance will decrease. Following the inverted-U model, the productivity of an agent can be calculated as a function of the stress, as shown in Equation 3.7, where S represents the stress.

$$P = \frac{1}{0.4 \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{1}{2}(\frac{S-0.5}{0.2})^2} \tag{3.7}$$

The above equation has been adjusted from the probability density function of a normal distribution $\mathcal{N}(0.5, 0.04)$. Productivity (P) will reach the maximum value of 1 when the stress level is 0.5, and will decrease as the value of stress moves away from its optimal value.

An agent with high productivity or performance is able to carry out more tasks in a shorter time span than an agent with a low level of performance. In that way, an agent with a non optimal level of stress may result in absenteeism, that refers to the number unscheduled leaves taken by an agent.

The diagram shown in Figure 3.2 represents an overview of the model design, showing the three components that compose stress along with the features that contribute to them.

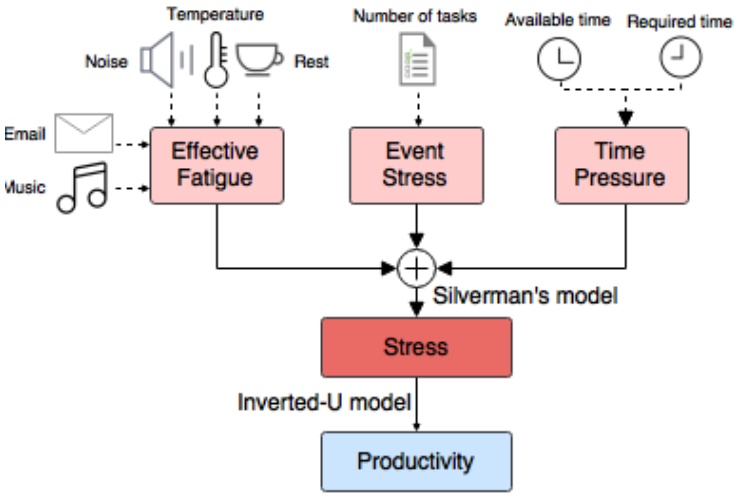

Figure 3.2: Stress model

3.2 Simulation system architecture

Once the theoretical aspects of the simulation have been explained, the next step is the development of the simulation system. The simulation system has been designed taking into account the requirements explained above, with the main purpose of developing a system that allows the representation of a smart office in a reliable way. The global architecture is shown in Figure 3.3.

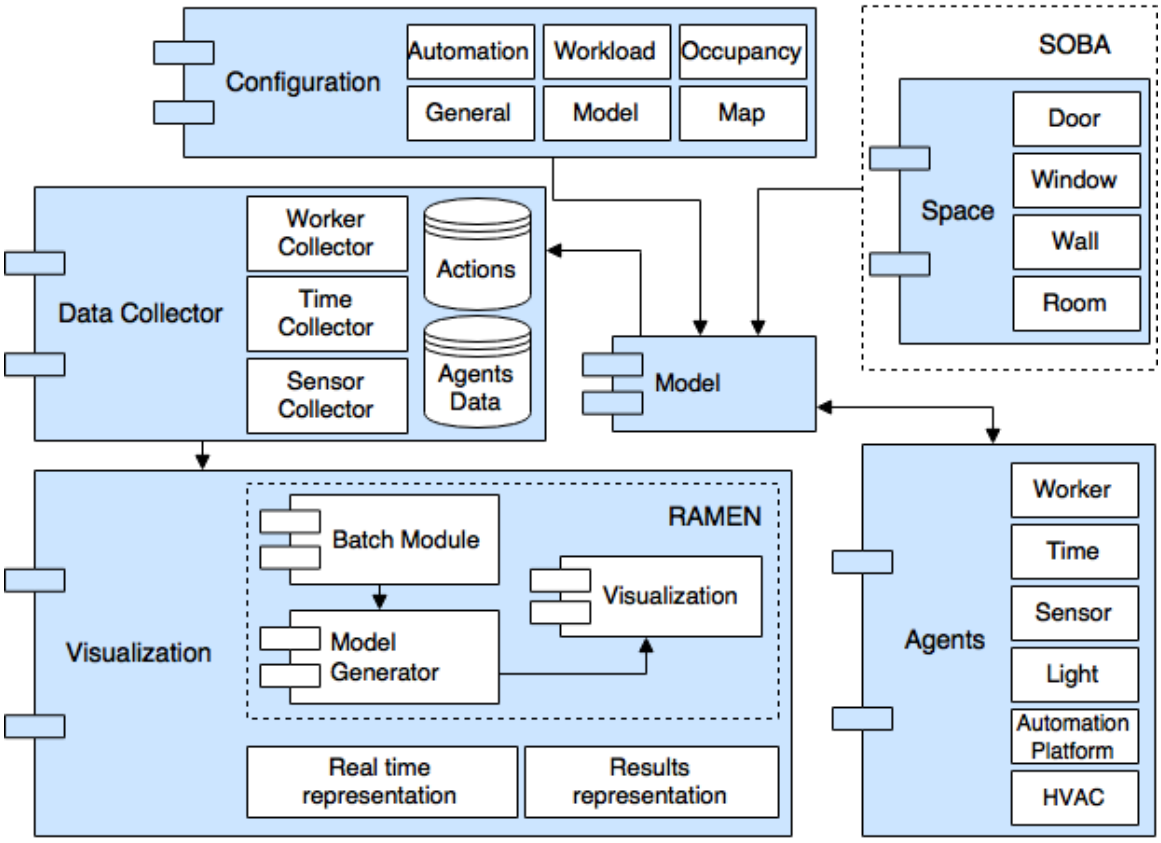

Figure 3.3: Simulation system architecture

As shown in the figure, the architecture consists of six main modules: *Model*, *Agents*, *Space*, *Data Manager*, *Visualization* and *Configuration*. In addition, it is integrated with two tools called SOBA and RAMEN, whose main features are explained in Section 2.1.4.

3.2.1 Model

The model represents the main class of the system, and it is an extension of MESA model. It is responsible for the creation of spaces and agents, as well as the management of all

the changes on them. For this reason, it connects with the *Space* and *Agents* modules. In addition, it also connects with the *Data Manager* module in order to collect different data related to the evaluation of the simulation.

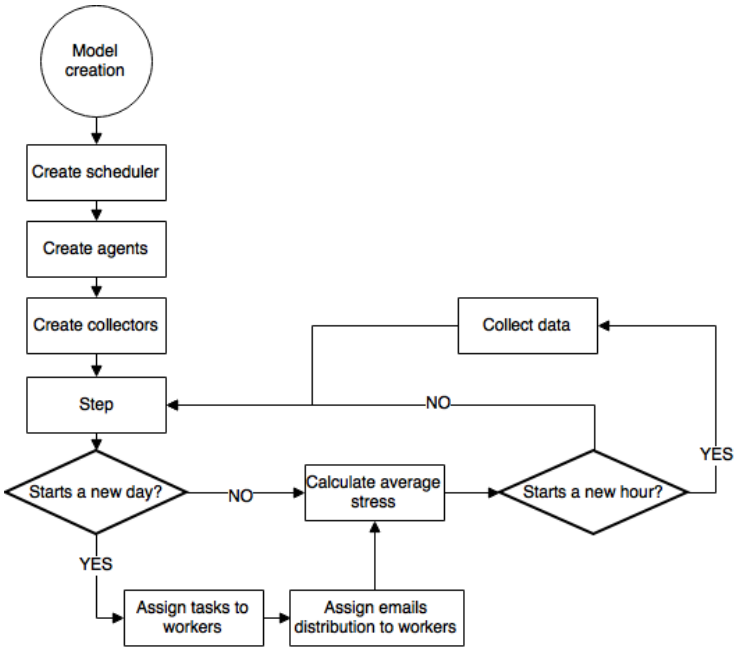

Figure 3.4: Model tasks flowchart

Figure 3.4 shows the tasks performed by the model during the simulation. When a simulation is started, the model is created, obtaining as unique argument the number of workers. Then, it creates the scheduler, the agents (and adds them to the scheduler), the building where they will be located, and the data collectors.

During the simulation performance, the model's step is the first to be executed, and then, the agents' schedule is run. At each step, the model checks in the timer agent if a new day has started. If it is a new day, the model will assign the tasks and the email reception distribution to each worker. The number of tasks and email distribution that assigns to each agent are calculated from two normal distributions whose values are specified at workload settings. Once the tasks have been assigned, the model calculates the event stress of each worker.

Then, the model calculates the average stress of workers, and lastly, checks if its a new hour. If the hour has changed, it collects the data related to the simulation by mean of the data collectors.

3.2.2 Agents

Several agent classes that extend the MESA Agent class have been defined in the system. These classes are: worker, time, sensor, light, HVAC and automation platform.

Worker. This class represents workers of the smart office environment, that are able to perform several actions such as work in tasks, read emails or rest. In addition, these agents may suffer from stress when their work or ambient conditions are not optimal, resulting in a decrease of the productivity. Figure 3.5 represents the tasks that performs the worker agent during the simulation.

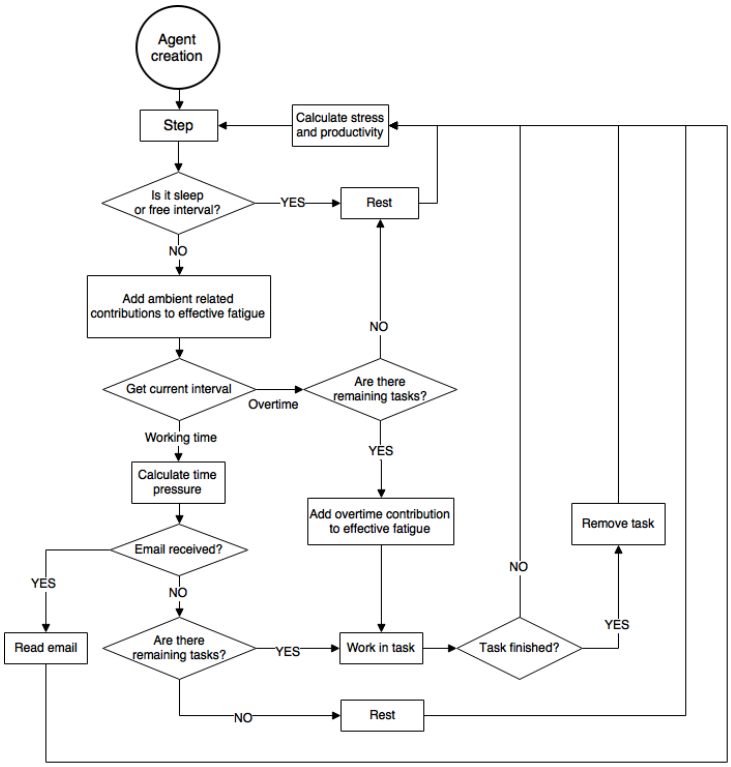

Figure 3.5: Worker tasks flowchart

As shown in the figure, at each step, the first is to get the current time interval. If it is sleep or free time, the worker will rest, and its effective fatigue will decrease; else, ambient conditions contributions (that depends on the values of temperature, humidity, luminosity, and noise) will be added to effective fatigue. If the current interval is overtime, the agent will check if there are remaining tasks, and will rest if there are not. However, if there are pending tasks, he will add overtime contribution to effective fatigue and will work in the task. If the current interval is working time, the time pressure will be calculated, and then will be checked if an email has been

received. If it has, the agent will read the email; else, he will work in a task. After working in a task, the worker checks if it has finished; and if it has, it is removed. Finally, the agent calculates new stress and productivity values.

Time. This class works together with the steps, which is key to achieve a correct performance with facets such as the worker's behavior or the ambient conditions modeling. Its function is to transform the simulations steps into time, in order to be able to have a control on days, hours, etc. At his step method, it converts the current step into days, hours and minutes; and defines if it is a new day or a new hour.

Sensor. This class represents an temperature, humidity, lighting and noise devices. These agents will be responsible for detecting different ambient parameters that are relevant to the stress estimation. Each sensor is associated to a room. At each step, the sensor measures these parameters depending of the states of HVAC and lights states.

Lighting. This class is used to represent each light of the environment, and it provides functions for managing the brightness of each light or controlling its state. Lights have two states: on and off; and different brightness values.

HVAC. The heating, ventilation and air conditioning system is modeled with this class. It provides functions for managing the state of each system and controlling in this way the temperature of the environment. At each step, it calculates the ambient temperature, that will be read from the sensor.

Automation Platform. Finally, this agent represents an automation platform, that can be configured for automating the control of ambient conditions, automating work related tasks, etc. This configuration is received from the automation configuration settings. At each step, this agent will adjust parameters of the simulation (such as graduate the light brightness, the HVAC power or the email reception) depending on the automation policy chosen. In this way, this agent is connected with all the rest.

3.2.3 Space

For modeling a physical space, the system has been integrated with SOBA, that composes the *Space* module of the system. This tool provides a module for modelling occupancy in buildings, enabling the definition of a physical space with interconnected rooms [52] as well as the movement of the workers in that space.

The modeling of the building has been designed considering that in a simulation for a smart office, the model could be simplified to work with occupants who make activities

in rooms associated to each activity or state. The movement between rooms implies a transition time for each movement corresponding to the sizes of each room and an average speed [89]. A worker agent moves between rooms depending on its stress level and its remaining tasks number (it will go to rest room if its stress level is high and will go back to work after the break). In addition, this module implements the A* search algorithm [90] for allowing workers agents to move in the building, between rooms, following the optimum way.

3.2.4 Visualization

In order to analyze the simulation results in an easier and more intuitive way, the system will be integrated with a visualization tool, called RAMEN. RAMEN is a tool based on Blueprint3D [2], that enables the visualization of the simulation (with all its agents, devices and places) in a 3D environment [53]. The integration of the simulation system with RAMEN makes simpler and faster to find solutions for possible problems or to reach conclusions from the simulation.

The simulation system will provide the needed files for defining the building with its rooms and devices, and once the simulation has been run, it will provide the movement file. This file declares the actions that occur during the simulation. It is divided in steps, a way of measuring time defined by the simulator, and for each step, it contains the needed information for representing all the agents involved in the simulation with their corresponding features. Some of the actions that can be executed are the following ones:

- Add new agent: it represents a new agent in the visualization. Required parameters: agent id (*agent*), initial position (*position*) and initial stress level (*stress*).
- Graduate lights: it enables to graduate the light level of a room. Required parameters: light level (*light*) and room where the light is located (*room*).
- Play music: it plays music on the TV. Required parameters: music state (*state*) and song (*song*) that must played.
- Move an agent: moves an agent to the selected room. Required parameters: agent id (*agent*) wanted to be moved, the room in which it is going to be moved (*room*) and the step in which the agent is going (*step*) to arrive to that position.

[2]Blueprint3D: https://github.com/furnishup/blueprint3d

Listing 3.1 shows an example of the movement file. It consists of two keys (*type* and *steps*). The former refers to the type of movement file, as RAMEN supports different formats for the movements; while the later refers to an array containing each step of the simulation. Each step object is also an array with all the actions that must be represented.

Listing 3.1: Movement file example

```
{
    "type" : 0,
    "steps": [
    [
        {
            "agent": 0,
            "position": "C.10",
            "stress": 0.5
        },
        {
            "agent": 1,
            "position": "C.1",
            "stress": 0.2
        }
    ],
    [
        {
            "light": "medium",
            "room": "Lab1"
        },
        {
            "state": "on",
            "song": "Relaxing"
        }
    ],
    [
        {
            "agent": 0,
            "moveTo": "C.2",
            "toStep": 15
        },
        {
            "agent": 1,
            "stress": "0.4",
        }
    ]]
}
```

In this example, two different agents are situated in the first step. Then, in the next step, the light of the Lab1 is set to medium level and the music player is turned on. Then, the movement of the agent 0 is declared, and the stress level of agent 1 is increased to 0.4.

In the visualization, the agent's color will change depending on the stress level, as shown in Figures 3.6, 3.7, 3.8, 3.9 and 3.10. The colors that represent each stress level are: light blue (lame stress level), dark blue (inattention stress level), green (optimal stress level), yellow (fatigue stress level) and red (anxiety stress level).

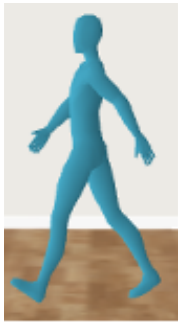

Figure 3.6: Lame

Figure 3.7: Inattention

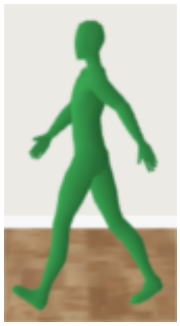

Figure 3.8: Optimal

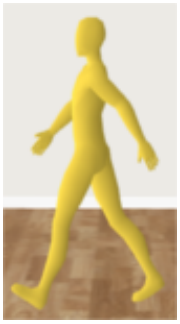

Figure 3.9: Fatigue

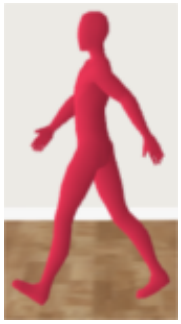

Figure 3.10: Anxiety

In addition to real time visualization, this module provides functions for representing the results obtained from the simulation. This functionality includes the representation of the following information in charts: average stress level evolution (also divided into its three components: event stress, effective fatigue and time pressure) of all workers, average productivity level evolution of all workers, stress and productivity evolution of a selected worker, final stress and productivity levels of all workers, relation between several ambient conditions and stress, average number of remaining tasks at the end of the day of all workers, etc.

3.2.5 Data Manager

This module is composed of three submodules, which are extensions of MESA's Data Collector. It is responsible for collecting and saving all the information related to the simulation, in order to enable weather the real time or subsequent analysis of the data and the extraction of results and conclusions.

The data will be stored in two repositories: *Action* repository and *Agents Data* repository. The former will contain information related to the real time visualization, including the movements of all workers agents along with their states, as well as the actions of the rest of agents (i.e. set the light brightness of working room to 60% or turn off air conditioning). The later, *Agents Data* repository, will contain useful information about certain parameters of the simulation that are not shown in the real time visualization and whose analysis results in an essential outcome of the simulation.

The model collector submodule collects average stress and average productivity values at each step. The worker collector collects data related to the worker (such as stress values with all its components, tasks completed or productivity). Finally, sensor collector collects data related to ambient conditions.

3.2.6 Configuration

The simulation is configured by different settings files. Two files define the building and the space of the environment: *occupancy* and *map*. The *occupancy* configuration file enables the modification of occupants activity and behavior, with their different states (working, resting, having lunch, etc.) and their schedule (arriving time, launch time, etc.), as well as the disposition of each agent at the building. The *map* file is used to define the building plan, with its rooms, walls, windows and doors.

With regards to work conditions, the file *workload* is used, and describes the task load for the workers, enabling the configuration of several parameters such as the average required time for each task, the tasks arrival distribution, emails reception distribution and the time spent in reading them, etc. The *general* file enables the specification of some general parameters related to the simulation, such as the number of workers or the equivalence in seconds of each step; and the *model* file defines the contributions of the different parameters to stress. Finally, the *automation* file contains the different automation policies that can be enabled or disabled.

The architecture explained above along with the designed model offer a complete sim-

ulation system that enables the realization of reliable simulations about a smart office environment. This allows the user the extraction of different types of information related to the environment and the workers in order to know the optimal ambient and work conditions that maximize workers productivity, as well as the best way to achieve these conditions by mean of the use of IoT technologies. In addition, the system has been designed with the purpose of being as configurable and scalable as possible, allowing the user to easily add new functions or components.

CHAPTER 4

Emotional Automation platform Architecture

In this chapter, the architecture of the emotion aware automation platform will be described, including the design phase and implementation details. The modules in which the whole system is structured as well as the interaction between modules or with the users will also be explained.

In the following sections, the architecture design of an emotion aware automation platform will be presented. The integration of emotion aware technologies in an IoT system opens endless possibilities, that are especially interesting in the application to a smart office scenario, as explained in Chapter 2.

The state of art analysis enables the extraction of a preliminary set of design criteria that have been followed when designing the architecture in order to achieve the goals of the project. These criteria result crucial for IoT systems whose purpose is the improvement of users' lives by mean of the automation of tasks. The following criteria have been identified:

1. *Scalability and interoperability.* Handling the growth of number of devices and information they produce is a massive challenge in IoT systems, as new devices and technologies are continuously being developed and its easy integration with current platforms results a key factor. In addition, the interaction amongst the great number of actors involved in these systems, comprising of human and non-human objects, is crucial to conceive the vision of IoT [91]. For this reason, semantic technologies have been integrated in the system, using Onyx ontology for the representation of emotions and EWE ontology for the representation of automation rules. Furthermore, automation rules will be evaluated by mean of EYE, a semantic rule engine.

2. *Real time processing capabilities.* Outdated information is of no use in real time applications, so the process of analysing and handling information has to be almost instantaneous. This supposes a major challenge for emotion recognition systems where the analysis to be done requires complex techniques that often spend too time on obtaining the results. The architecture proposed will integrate real time emotion recognition by mean of speech and face analysis.

3. *Non intrusive methods.* Continuous and non intrusive monitoring of various parameters of a human or an environment is a great challenge for IoT systems. This is particularly crucial in a smart office context, where the system is developed with the main purpose of maximizing the productivity of workers. Thus, the architecture has been designed in order to minimizing both the invasiveness of sensors and the need of user interaction.

4. *Accuracy.* When analysing and detecting emotions, the accuracy obtained by the system results a crucial factor. In this project, the correct behaviour of the system depends on the correct recognition of emotions, so much effort has been done for improving this task in the architecture design.

Once the design criteria has been described, the architecture proposed will be explained in detail. The system is divided into two main blocks: **emotion recogniser** and **automation platform**, as shown in Figure 4.1.

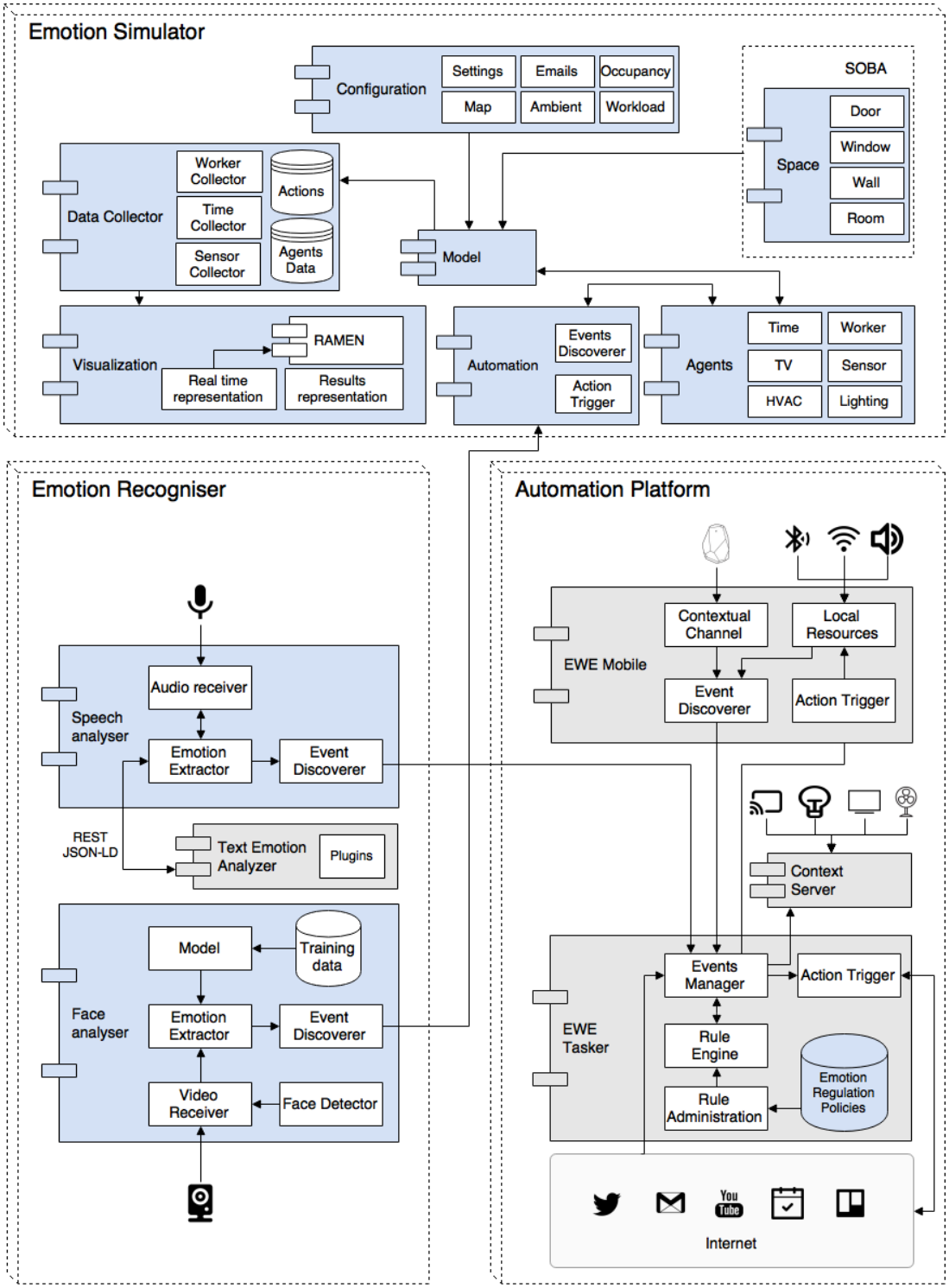

Figure 4.1: Emotion aware automation platform architecture

The former is composed of the modules involved in the emotion recognition of the user, that can be extracted in two ways: speech analysis and face recognition; while the later is composed of the modules that enable the automation of tasks. This master thesis focuses on the development of the emotion recogniser (represented with blue in the Figure), as the automation platform will be based on a previous work called EWE Tasker [92].

In addition, the simulation system explained in the previous chapter has been integrated. In order to carry out this integration, a new module has been developed: *Automation*, that is composed of an *events discoverer* that connects with the agents to receive information that will use for generating events and send them to the automation platform; and an *action trigger*, that connects with the agents for performing the subsequent actions.

Emotion recogniser aims to detect and recognise users' emotions and send them to the automation platform in order to trigger the corresponding actions. It consists of two modules: speech analyser and face analyser, and it is integrated with Senpy[1], that provides emotion extraction from text.

As commented above, the emotions extracted are sent to the automation system, a semantic event-driven system platform that receives events from several sources and performs the corresponding actions. In addition, it provides several functions for automating tasks by means of semantic rules and integrates different devices and services.

4.1 Emotion recogniser

The emotion recogniser block is responsible for providing functions that enable the detection of users' emotions, its representation using semantic technologies and the sending of this data to the automation platform, where it will be evaluated. The block is composed of two main modules, face analyser and speech analyser. In addition, speech analyser is integrated with Senpy, the Text Emotion Analyzer module, described in Section 2.1.5.2, for extracting the emotions of the text.

Both modules are Python applications, in order to maximize the compatibility of the modules with any current computer. In addition, each module is composed of multiple submodules that provide the required functions, with the purpose of making the system easy to handling. In this way, speech face analysers are divided into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.

[1]Senpy: http://senpy.cluster.gsi.dit.upm.es

Face analyser. The main goal of this module is the extraction of users' emotions by mean of real time recognition of facial expression, carried out with Convolutional Neural Networks (CNN). It is responsible for providing functions that enables the connection with the webcam for obtaining the user's images that are the inputs of the deep learning model, the processing of these images for obtaining the features that enable the extraction of emotions, the modeling of these emotions with semantic technologies and the sending of these emotions to the automation platform. In addition, it provides functions for training the deep learning model, in order allow the user to use his own model.

Speech analyser. This module has been developed with the purpose of obtaining users' emotions from their speeches. Thus, it provides functions for connecting with the microphone in order to get the user's speech, transcribing it to text, connecting with Senpy module for obtaining the emotions of the text, and, in the same way that face analyser, modeling these emotions with semantic technologies and sending of these emotions to the automation platform.

Once an overview of the block has been given, down below both modules along with their respective submodules will be described in detail, explaining their main features and design aspects.

4.1.1 Face analyser

This module is divided into five submodules: *video receiver*, *face detector*, *model*, *emotion extractor* and *event discoverer*; each of them with different functions that help to achieve the goal of the face analyser module: detect emotions from users' faces.

Before explaining these submodules in detail, it is interesting to briefly describe some of the libraries that have been used in the development of the module, with the purpose of clarifying and making easier to understand the functioning of each submodule.

OpenCV. Open Source Computer Vision Library[2] is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. This library, with Python support, provides several features that result particularly interesting in this thesis, such as image processing, video analysis, and an easy-to-use interface to video capturing and video codecs.

[2]OpenCV: http://opencv.org

Keras. Designed to enable fast experimentation with deep neural networks, Keras[3] is a high-level wrapper of Theano and Tensorflow. It provides friendly APIs to manipulate several kinds of deep learning models, focusing on being minimal, modular and extensible.

Scikit-learn. A free software machine learning library[4] designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN.

H5PY. This library is a Pythonic interface to the HDF5[5] binary data format, that lets you store huge amounts of numerical data, and easily manipulate that data from NumPy. HDF5 supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data.

The integration and use of these libraries in the different submodules will be explained in the following sections, along with the implementation and design details of each submodule.

4.1.1.1 Video receiver

In order to perform real time recognition of facial expression for detecting emotions, the developing of a module that handles the video obtaining results crucial. That is the main goal of this submodule: capturing video from the camera.

With this purpose, video receiver captures video from the camera by mean of OpenCV functions. The process is quite simple as shown in the flow chart represented in Figure 4.2, thanks to the use of OpenCV library. The first step is to get the video streaming by mean of the creation of an OpenCV VideoCapture object. After that, a window for showing the captured video on the user's screen will be created or not depending on user's configuration. Once created the Video Capture object, it is possible to capture frame by frame using the *read* method of the VideoCapture object, and show in the window the image obtained.

However, before being able to extract the emotion, it is necessary to carry out some preprocessing of the image. This preprocessing is done by the face detector submodule, so the frames captured by video receiver will be passed to it.

[3]Keras: https://keras.io
[4]Scikit-learn: http://scikit-learn.org/
[5]HDF5: https://support.hdfgroup.org/HDF5/

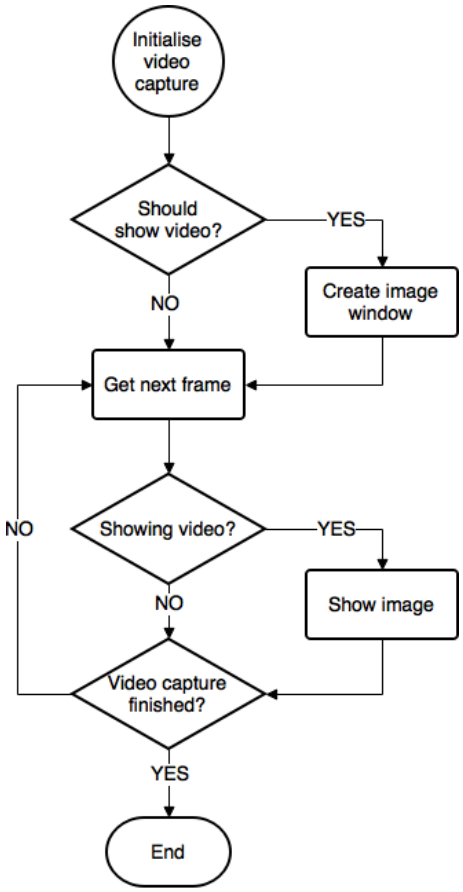

Figure 4.2: Video capture flowchart

4.1.1.2 Face detector

Once the video streaming is being obtained, each frame captured has to be preprocessed for enabling the subsequent detection of the emotion. The preprocessing of each frame is the main purpose of the face detector submodule. In the same way that video receiver, this submodule uses OpenCV functions for getting the coordinates of the face and adapt it to the required dimensions. This process, shown in Figure 4.3 is not straightforward, and will be explained below.

The detection of the face consists of several tasks. The first step is to equalize the histogram of the image. Image histogram refers to the graphical representation of the intensity distribution of an image, and quantifies the number of pixels for each intensity value considered. The histogram equalization improves the contrast in an image, in order to stretch out the intensity range and facilitate the face detection. However, the *equalizeHist* function of OpenCV works with grey scale images, so previously the color space must have been converted to this space. This conversion also makes processing images much, much

faster. Finally, a cascade classifier will be applied in order to detect the faces of the image. OpenCV already contains many pre-trained classifiers for face, eyes, smile, etc[6]. In this case, *haarcascade_frontalface_default.xml* has been used, as it contains a ton of features that have been decided as belonging to a front-facing face. The *detectMultiscale* function of the classifier returns the face coordinates.

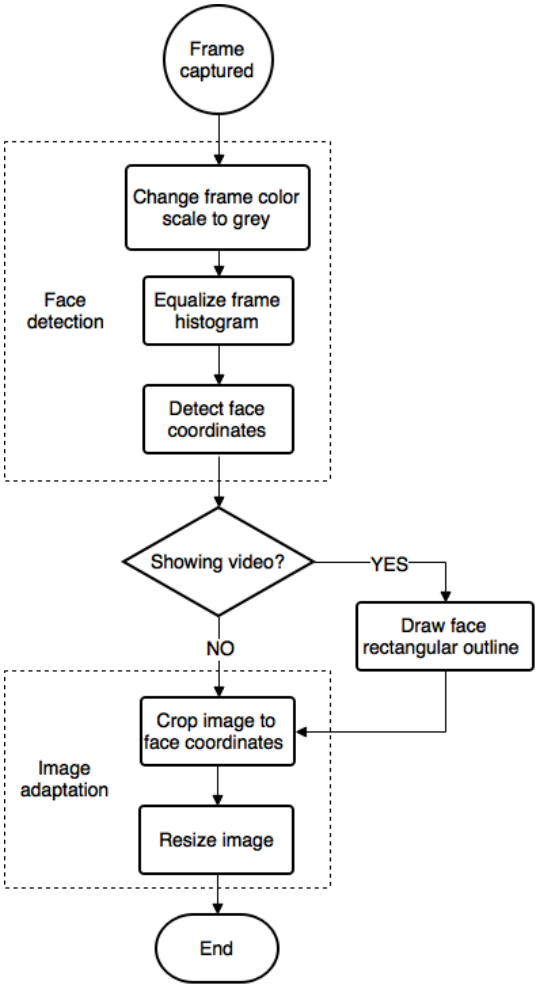

Figure 4.3: Image preprocessing flowchart

In addition, the face will be extracted from the original image and redimensioned to the required shape, in order to be similar to the training images of the model.

4.1.1.3 Model

The model is a Python implementation for Keras of an adaptation of VGG16 model. This model, proposed by Karen Simonyan and Andrew Zisserman [93], consists of a 16-layer net-

[6]OpenCV cascade classifiers: https://github.com/opencv/opencv/tree/master/data/haarcascades

work that achieves 7.5% top-5 error on ILSVRC-2012-val and 7.4% top-5 error on ILSVRC-2012-test.

Original VGG16 model is composed of a stack of 13 convolutional layers, where filters with a very small receptive field (3×3) are used. The convolution stride, that controls how much the filter shifts at each step, is fixed to 1 pixel; and the spatial padding of convolutional layer input, that provides control of the output volume spatial size, is such that the spatial resolution is preserved after convolution. Some of these convolutional layers are followed by a max-pooling layer in order to carry out spatial pooling (a form of non-linear down-sampling), which is performed over a 2×2 pixel window with stride 2. Max-pooling layers partition the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. In addition, the stack of convolutional layers is followed by three Fully Connected (FC) layers that do the high-level reasoning: the first two have 4096 channels each and the third has 1000. The final layer is the soft-max layer. All hidden layers are equipped with the rectification (ReLU) non-linearity, proposed by Krizhevsky et al. [94], that applies the non-saturating activation function in order to increase the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

However, the model used in this thesis is an adaptation that has some modifications with regard to the original model. This adaptation has been carried out because the original model was optimized for receiving RGB images with a fixed-size 224×224, while the input images of this system will be 48 x 48 gray scale. Specifically, the number of filters at each convolutional layer has been reduced, as well as the number of channels of FC layers (that will have 1024, 512 and 6 respectively). In this way, 7 convolutional layers are used instead of 13. Figure 4.4 shows the representation of the CNN architecture.

The implementation is based on a sequential keras model, which is a linear stack of neural network layers. Keras sequential model provides functions for easily adding the necessary layers, as described in Appendix D. Once the model is created, the next step is to define the loss function and the optimizer that will be required when training the model (finding the best set of weights to make predictions). The loss function is used to evaluate a set of weights, while the optimizer is used to search through different weights for the network and any optional metrics which are interesting to collect and report during training.

The loss function chosen is *categorial crossentropy*, that computes the categorical cross-entropy between predictions and targets; while the optimizer chosen is *adam*, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [95]. Finally, because it is a classification problem,

the classification accuracy will be collected and reported as the metric.

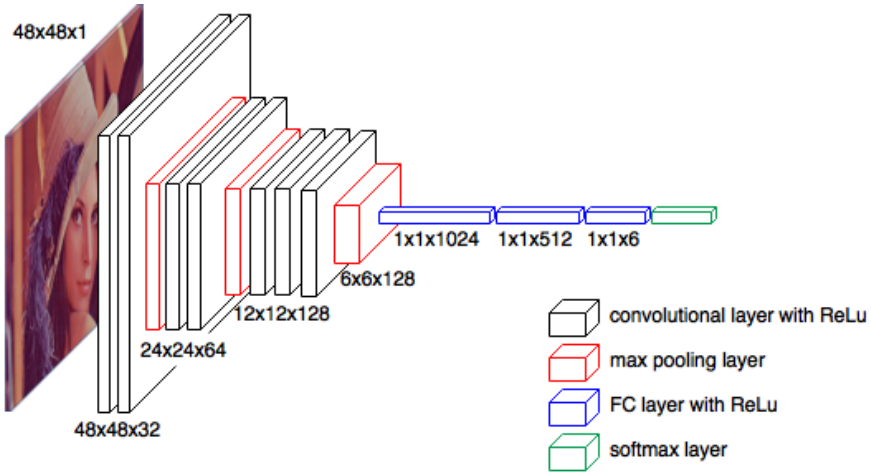

Figure 4.4: Convolutional Neural Network model architecture

In addition, the model has to be trained, so a function with this purpose has been developed. The training data has been extracted from fer2013, anounced in Kaggle competition in 2013. This dataset consists of 28.709 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is centered and occupies about the same amount of space in each image [96]. The training function uses the *fit* function of OpenCV for obtaining the best set of weights, that will be stored in h5 format for its subsequent use.

4.1.1.4 Emotion extractor

Once the model has been created and trained, and the face has been detected in a frame, it is possible to extract the emotion of the face. This task is performed by this module, the emotion extractor, that uses the *predict* function of the keras model for getting the user's emotion. This function generates output predictions for the input samples.

However, before calling the *predict* function, it is necessary to modify the dimensions of the input, as the function expects some array of samples and in this case only an image is passed. The input should be 4-d, with the 1st dimension to enumerate the samples, so as the image has two dimensions, it is necessary to add two more by mean of *expand_dims* function of NumPy library..

The obtained result is an array of containing the probabilities for each emotion, so the emotion with maximum value is extracted, completing the emotion detection. Nevertheless, in order to provide more accuracy to the emotion detection, instead of taking as valid each detected emotion, this submodule takes the most detected emotion in a time span that can

be configurable by the user. In addition, only predicted emotion with a probability greater than 75% are taken into account.

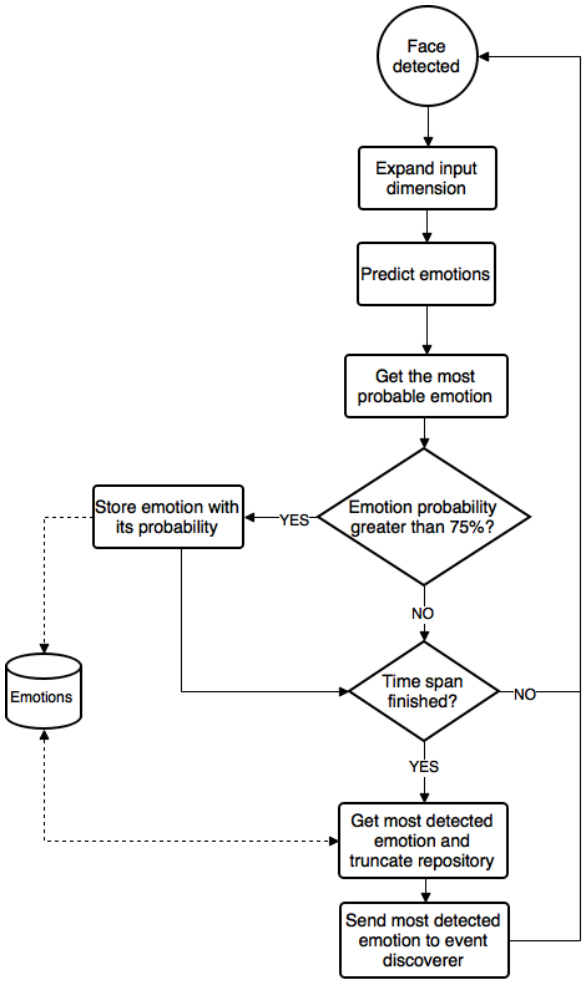

Figure 4.5: Emotion extraction flowchart

The most detected emotion of a time span is passed to the event discoverer where will be semantically modeled and sent to the automation platform. Currently, six emotions are detected: neutral, anger, sadness, happiness, surprise and fear.

4.1.1.5 Event discoverer

This module receives the detected emotion from the *emotion extractor*, and its main role is to apply a semantic layer to the emotion detection, and send it to the automation platform.

As explained in Section 4.2, the automation platform receives events from different channels, that trigger actions depending on the evaluation of these events along with rules by a semantic rule engine. Thus, *event discoverer* must act as an event source, generating

events from the emotion detection and sending them to the automation platform.

Emotions are represented with Onyx ontology, explained in Section 2.1.2.1, while events are modeled using EWE, explained in Section 2.2.4. Listing 4.1 shows an example of an event written in Notation3. The event has been generated by the detection of sadness with a probability of 97%.

Listing 4.1: Event generated by the detection of an emotion by face analyser

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ewe-emodet: <http://gsi.dit.upm.es/ontologies/ewe-emodet/> .
@prefix ewe: <http://gsi.dit.upm.es/ontologies/ewe/ns/#> .
@prefix ex: <http://example.org/> .
@prefix onyx: <http://www.gsi.dit.upm.es/ontologies/onyx/> .
@prefix wn-affect: <http://gsi.dit.upm.es/ontologies/wnaffect/ns#> .

ex:EmotionDetected rdf:type ewe-emodet:EmotionDetected.
ex:EmotionDetected ewe-emodet:detectedBy ewe-emodet:FaceAnalyser.
ex:EmotionDetected ewe-emodet:hasDetected onyx:Emotion.
onyx:Emotion onyx:hasEmotionCategory wn-affect:Sadness .
onyx:Emotion onyx:hasEmotionIntensity 0.97 .
```

As shown in the listing, the event specifies which analyser has detected the emotion (face or speech). This allows the user to generate rules depending on the source of the detection, or even include both analysers for increasing the reliability of the detection.

The event is sent to the automation platform via HTTP request, where will be evaluated along with the rules predefined by the user, triggering the corresponding actions.

In order to sum up and clarify the architecture of the face analyser module, the flowchart represented in Figure 4.6 shows an overview of the whole detection process, focusing on the most relevant tasks.

As shown, the process starts with the video capture initialization. Then the model is created and trained (if it has not been trained before). At the same time, the current frame is captured, and starts the proccess of detecting the face and adapt the image to the required dimensions. Then, the emotion is detected, and the emotion detection event is generated and sent to the automation platform. This process will iterate until the video capture finishes.

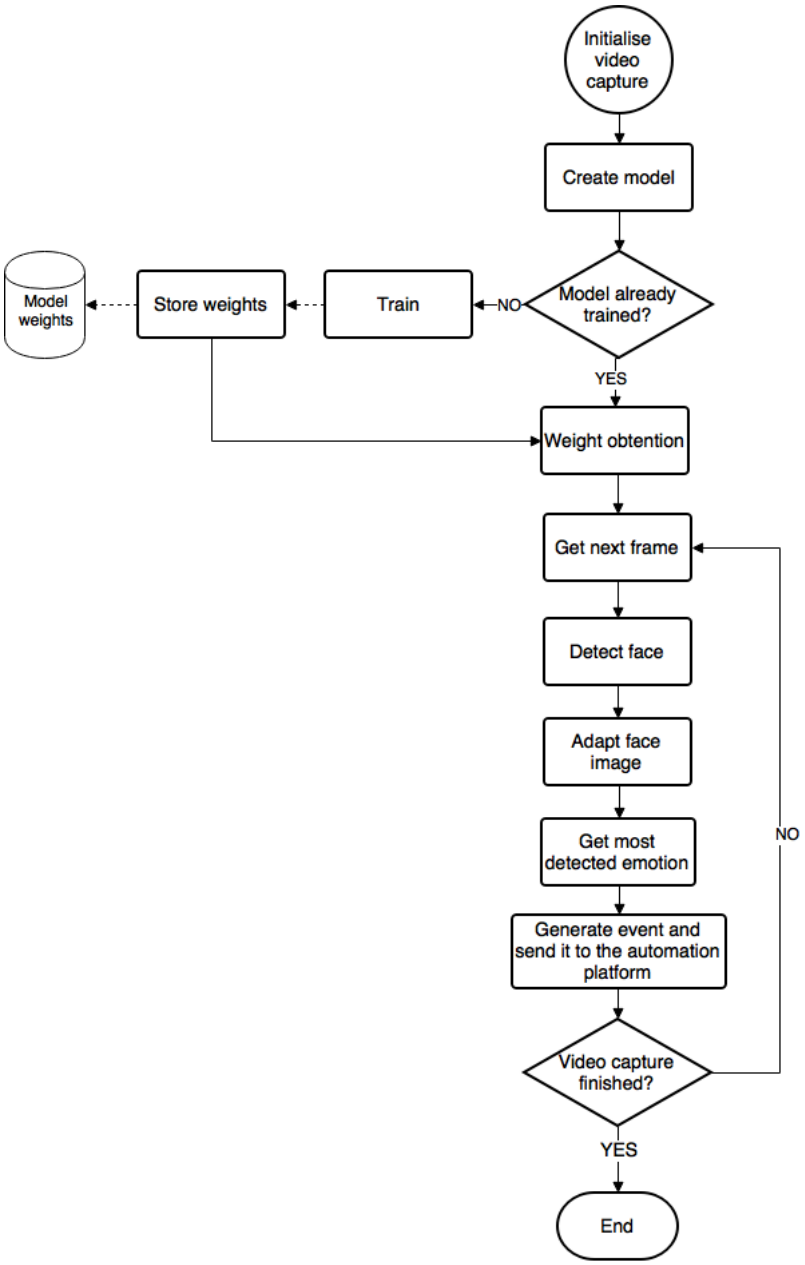

Figure 4.6: Whole emotion detection by face recognition process flowchart

4.1.2 Speech analyser

Speech analyser is responsible for detecting emotions from speech recognition, and with the purpose of achieving this goal, is divided into three submodules: audio receiver, emotion extractor and event discoverer. In addition, this module is connected with Senpy tool, in order to perform emotion recognition from text.

It is important to briefly describe some of the libraries that are used in the implemen-

tation of this module, in order to make easier to understand the functioning of the different submodules involved in the emotion detection from speech.

PyAudio. This library provides Python bindings for PortAudio[7], the cross-platform audio I/O library. PyAudio[8] enables the use of Python to play and record audio on a variety of platforms in an easy way.

SpeechRecognition. It enables performing speech recognition, with support for several engines and APIs, online and offline. SpeechRecognition[9] provides a high level interface for working with different APIs such as Google Cloud Speech API, Microsoft Bing Voice Recognition and IBM Speech to Text. In this thesis, it will be used with Google Cloud Speech API.

Google Cloud Speech API. It enables the conversion of audio to text by applying powerful neural network models in an easy to use API. It recognizes over 80 languages and variants. The Speech API provide functions for sending audio and receiving a text transcription from the service[10].

Down below, the integration and use of these libraries in the different submodules will be explained, along with the implementation and design details of each submodule.

4.1.3 Audio receiver

Audio receiver is responsible for connecting with the microphone in order to obtain the speech that will be used for extracting the emotion, and for transcribing the voice to text. The connection with the microphone is done by mean of PyAudio and SpeechRecognition libraries. These libraries make the audio reception process and its subsequent transcription quite simple, as shown in Figure 4.7.

As the audio reception will be running in the background, the first step before starting to listen is to realize and adjustment of ambient noise, with the purpose of increasing the accuracy of the detection. SpeechRecognition provides a function for doing this: *adjust_for_ambient_noise*, that receives as argument the microphone object. This calibration is only needed before starting to listen. Once calibrated the ambient noise, the voice recognition can be initialized by mean of *listen_in_background* function of SpeechRecognition

[7]PortAudio: http://www.portaudio.com
[8]PyAudio: https://people.csail.mit.edu/hubert/pyaudio/
[9]SpeechRecognition: https://pypi.python.org/pypi/SpeechRecognition/
[10]Google Cloud Speech APIhttps://cloud.google.com/speech/

library, and it will be running in the background waiting for receiving audio data. When data is received, this module is responsible for transcribing it to text, in order to enable the emotion extraction from it by mean of Senpy tool. For doing this task, audio receiver module uses the functions provided by SpeechRecognition library that enable the connection with Google Cloud Speech API.

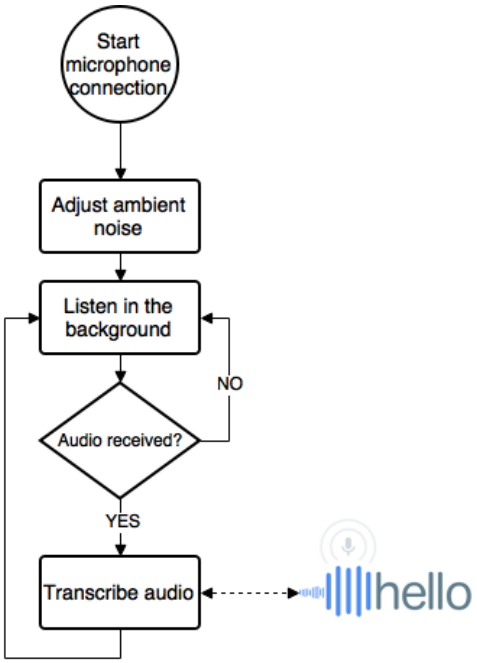

Figure 4.7: Audio reception process flowchart

Specifically, the function used is *recognize_google*, that connects with Google Cloud Speech API, sends the recorded audio and obtains the text as response. Once the transcription of the speech has been done, it must be passed to Senpy, where will be analysed. *Emotion extractor* submodule is responsible for this task.

4.1.4 Emotion extractor

This submodule manages the communication with Senpy analysis platform. It receives the transcription of the audio from the *audio receiver*, creates the request that sends to Senpy, and parses the response for obtaining the emotion. The communication between emotion extractor and Senpy is done via HTTP request.

As explained in Section 2.1.5.2, Senpy has different plugins that can be used for obtaining emotions and sentiments of a text. Emotion extractor will use *EmoTextANEW*, as it provides the best results for this context. The plugin used must be included in the GET

request to Senpy API, along with the text to be analysed.

The response given by Senpy is a JSON that contains different parameters. For each text, it calculates three values: arousal, dominance and valence. For calculating the emotion, Senpy compares the calculated values with a static centroids that describe these parameter values for several emotion states, in order to calculate the nearest emotion for each audio. Each of these centroids are represented in Table 4.1, with its corresponding arousal, dominance and valence reference values. The acquired emotion is also sent in the response.

Emotion	**Arousal**	**Dominance**	**Valence**
Anger	6.95	5.1	2.7
Disgust	5.3	8.05	2.7
Fear	6.5	3.6	3.2
Joy	7.22	6.28	8.6
Sadness	5.21	2.82	2.21

Table 4.1: Emotions centroids representation

After receiving the emotion from Senpy, emotion extractor sends it to the events discoverer submodule, that will be responsible for applying the semantic layer and sending the event of emotion detection to the automation platform.

4.1.4.1 Event discoverer

This module acts in the same way that its counterpart of face analyser module explained in Section 4.1.1.5, so it will not be described in detail. As shown in Listing 4.2, the main difference is the change in the detector, that now is speech analyser, and the lack of emotion intensity property, as it is not provided by the Senpy plugin.

In the same way that in the case of event discoverer of face analyser module, the event is sent to the automation platform via HTTP request, where will be evaluated along with the rules predefined by the user, triggering the corresponding actions.

In this way, the user will be able to configure rules that depend on the analyzer that has detected the emotion, although the most common use is to require the detection by speech analyser and face analyser, in order to improve the accuracy of the detection.

Listing 4.2: Event generated by the detection of an emotion by speech analyser

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ewe-emodet: <http://gsi.dit.upm.es/ontologies/ewe-emodet/> .
@prefix ewe: <http://gsi.dit.upm.es/ontologies/ewe/ns/#> .
@prefix ex: <http://example.org/> .
@prefix onyx: <http://www.gsi.dit.upm.es/ontologies/onyx/> .
@prefix wn-affect: <http://gsi.dit.upm.es/ontologies/wnaffect/ns#> .

ex:EmotionDetected rdf:type ewe-emodet:EmotionDetected.
ex:EmotionDetected ewe-emodet:detectedBy ewe-emodet:SpeechAnalyser.
ex:EmotionDetected ewe-emodet:hasDetected onyx:Emotion.
onyx:Emotion onyx:hasEmotionCategory wn-affect:Joy .
```

This section concludes with a summary of the emotion detection process from speech, with the purpose of clarifying the architecture of the module. The flowchart represented in Figure 4.8 shows an overview of the whole detection process, focusing on the most relevant tasks.

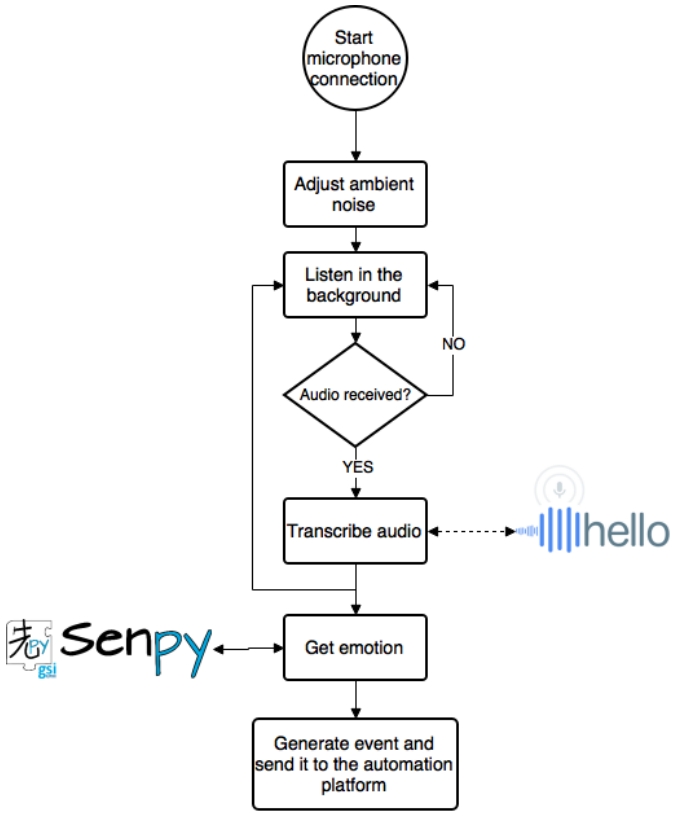

Figure 4.8: Whole emotion detection by speech recognition process flowchart

As shown, the process starts with the connection with the microphone. Then the ambient noise is adjusted in order to increase the transcription accuracy, and once that noise has been adjusted, the module starts to listen in the background, waiting for any audio data. When audio is received, it is transcribed and the resulting text is sent to Senpy tool. This tool will respond with the emotion analysis result, and when this happens, the emotion detection event is generated and sent to the automation platform. This process will iterate until the audio listening is stopped.

4.2 Automation platform

As commented above, the task automation platform is based on a previous work named EWE Tasker. Although the full details of this platform are given in [92], it is necessary to provide a brief overview of the block in order to successfully understand the whole system. EWE Tasker is an intelligent automation platform based on semantic ECA (Event-Condition-Action) rules, developed in 2016. Its main goal is to enable semantic rule automation in a smart environment, allowing the user to configure his own automation rules or to import rules created by other users in an easy way. In addition, it provides integration with several devices and services such as iBeacons, smart TV, Twitter, Github, etc., as well as an easy way for carrying out new integrations.

It is composed of three main modules: **EWE Tasker core**, **EWE Mobile** and **Context server**. The former is the core of the system, and serves as a platform that handles events coming from different sources and triggers accordingly the corresponding actions generated by the rule engine, but also includes all the functions for managing automation rules, creating and editing channels, and the repositories where rules are stored. EWE Mobile connects with different contextual channels (such as iBeacons) or local resources (such as bluetooth, wifi, battery level) for receiving information and generate events that are sent them to the EWE Tasker core or for triggering actions (like muting the mobile phone). In addition, it also provides functions for managing rules from the mobile. Finally, context server module acts an intermediary that connects with different channels which are not in the same network that the core (such as Chromecast or smart lights), and enables the trigger of actions as well as the receiving of events from them.

EWE Tasker Core. Users in a smart environment need a platform to manage their automation rules. Such a platform must provide functions for creating, updating, storing and erasing these rules. In addition, when programming these automations, users need to activate available channels. With the purpose of making easier the

process of plugging new components, EWE Tasker core provides functions for also managing these channels. Finally, the task automation platform has a main role: to automate tasks. It must be able to conect with several channels for receiving events, evaluating them together with stored rules and performing the corresponding actions. EWE Tasker Core module has been designed to carry out all these functions, and it has been split into four submodules.

Rule Administration provides an automation rule editor in which users can configure and adapt their preferences about events coming from all sources. Users are able to create, remove, or edit rules in an easy way by a rule editor graphical interface, based on icons and "drag and drop" actions.

Channel Administration provides users facilities for creating channels. Actions and events are defined using Notation3. Once the required events and actions have been created, channels can be defined properly. In addition, this module has an *events manager* submodule that is responsible for handling events coming from all sources, and send them to the *rule engine*.

Rule Engine is one of the most important modules in this system, and is based on an ontology model, which uses the EWE ontology. It is divided into two parts: EYE Helper and EYE Server. The former is responsible for the reception of events from the *channel administration* and the load of rules that are stored in the repository. When a new event is received, EYE Helper captures it and loads the available rules. Then, events and rules are sent to the EYE Server, an Euler Yap Engine reasoner that runs the ontology model inferences. The rule engine draws the actions based on the incoming events and the automation rules, and after performing some processing, sends them to the *action trigger*.

Action Trigger has as main role to trigger the actions generated by the *rule engine*. The Action Trigger of the EWE Tasker core is able to run actions that come from several channels like: Trello or Google Calendar. For this purpose, the Action Trigger module must be connected with these channels.

To sum up, automation rules are created in the *rule administration* using channels created in the *channels administration*. This submodule also receives events that are evaluated together with stored rules in the *rule engine*. Once the consequently actions have been generated, they are triggered in the *action trigger* module. As described above, the EWE Tasker core is able to handle automation rules from their creation to their execution.

EWE Mobile. This module is an event source that receive information from several

sensors such as iBeacons or the mobile device itself, and generates events coming that are sent to the EWE Tasker core. In addition, it provides functions for managing rules from the mobile.

Contextual Channel receives information from physical devices and sensors, such as temperature, humidity or presence, and sends this data to the event discoverer, that will transform it into Notation3 events that will be sent to the core.

Local resources submodule connects with mobile device features in order to obtain several information such as battery level, wifi connection state or sound level and sends it to the *event discoverer*, where will be tranformed into events and passed to the core.

Event discoverer is responsible for converting the information received from different sources to semantic Notation3 events, that will be evaluated by the *rule engine* in the core.

Action trigger submodule has the same goal that its counterpart in the core: to trigger actions. For this reason, it connects with several contextual devices and sensors as well as with components of the mobile device itself.

This thesis proposes some changes in this module in order to being able to receive temperature and humidity values from iBeacons, as it is an interesting feature for the smart office context that was not fully developed on previous work. In addition, functionality for detecting ambient light level as been also implemented, as it is a useful event source for the context.

Context server. This module has as main role the connection with several channels that are not accessible from the external network where the core is located, in order to be able to receive information from them for generating events and perform actions in which they are involved. It can connect with devices located in a local network such as a smart tv, an air conditioning system, a connected door, chromecast device, etc. In this thesis, the connection with a sound level meter has been implemented, in order to obtain the level of noise at the smart office and use it as an event source.

In addition, a contribution of this thesis to the EWE Tasker automation platform is the integration of emotion regulation policies. This policies are set of rules which aims to regulate the emotion intensity in different contexts. In the smart office context proposed in this thesis, they are intended to regulate negative emotions in order to maximize productivity.

These rules may be related to automate ambient conditions control for improving the workers' comfort, to automate work related tasks for improving the efficiency or to adjust

work conditions for improving productivity. Some examples of these rules are presented below:

(a) *If stress level of a worker is too high, then reduce his task number.* When a very high stress level in a worker has been detected, this rule proposes reducing his workload in order to achieve that his stress level falls and his productivity rises.

(b) *If temperature rises above 30^{o}C, then turn on the air conditioning.* To work at high level of temperatures may result in workers' stress as described in Chapter 3, so this rule proposes to automatically control this temperature in order to prevent high levels of stress.

(c) *If average stress level of workers is too high, then play relaxing music.* If the most of workers have a high stress value, the company productivity will be significantly fall. So this rule proposes to play relaxing music in order to reduce the stress level of workers.

Emotion regulation policies are stored in a EWE Tasker Core repository, and can be evaluated with the simulation system or implemented in the real scenario. In addition, company managers may implement their own emotion regulation policies in order to adjust the system to their own context.

To conclude the chapter, an overview of the whole architecture is presented. The whole system is divided into two big blocks with different functions: emotion recogniser, which aims to detect emotions from speech and face recognition and the automation platform, that aims to perform task automation and to allow the user to configure his own rules. In this way, emotion recogniser block will acts as a channel that generates events related to the detection of emotions and sends them to the automation platform, where they are evaluated along with the user's automation rules in order to carry out the task automation. Furthermore, the automation platform has been integrated with the simulation system described at the previous chapter, in order to allow the simulation and test of rules. Finally, some emotion regulation policies have been defined, which aims to regulate emotions increasing the intensity of positive ones and decreasing the intensity of the negatives.

CHAPTER 5

Case Study

In this chapter, a complete case study of the developed system will be presented. The application of the system to a living lab as well as its application to a social simulated scenario will be extensively described, explaining the usage of all the tools involved and their purpose, as well as the evaluation of the system.

To demonstrate the capabilities of the system proposed in the previous chapters and to help in the understanding of the project functionalities, the following sections present its application to a real case, covering the main features of the system.

The proposed scenario consists in a technology company that wants to implement an automation platform at its office, in order to increase workers productivity and improve the work conditions. However, before implementing the platform in a real scenario they are interested in analysing their environment in order to have a vision of the most important automations to implement. With this purpose, they want to perform some social simulations in a smart office environment with emotional agents, for providing more reliability to the analysis.

5.1 Simulation

The company defines three strategies or policies to be evaluated: Baseline, Ambient Automation and Workload Automation, in order to analyse the optimum automation level that must implement at the office. The strategies have the following features:

Baseline. It is used as the reference of the current method of operation in the office, and represents the situation to improve. There is not any task automation nor log system to detect workers' or ambient parameters, and the equipment (such as tv, HVAC system and lights) is controlled by the workers, who manually turn on, turn off and configure all the devices. In this way, ambient conditions such as noise, temperature, humidity or luminosity may not have optimum values; decreasing work conditions and affecting to workers' mood and productivity.

Ambient Automation. This strategy represents an environment where automation of ambient conditions control has been implemented. In this way, ambient conditions will always be near optimum values, and the work conditions will be significantly improved.

Workload Automation. The environment represented by this policy is a whole automated office, where in addition to ambient, automation of workload related tasks has been automated. In this way, workers can configure rules that automate some of their daily tasks, which turns out in an relevant time saving.

The goal of the company is to analyse if the implementation of the automation platform really translates into an increase of workers' productivity and an improvement of the

work conditions, to see if it is really worth to invest in the implementation of some of the automation policies.

Once the different strategies that will be evaluated have been described, the simulation process will be explained. The scenario is an office with 10 workers that work divided into two rooms. In addition, they have a rest room where they have a break. Workers have an schedule with the arriving time to work, the leaving time, the overtime hour limit, the free time and the sleep time. In this way, the activity of workers is divided into four intervals: working, working overtime hours, free time, and sleep time. In the *working* interval (from 9 a.m. to 5 p.m.), workers will work on the tasks that have been assigned to each one and reading received emails. If they do not complete these tasks before leaving time, they will work on them in the overtime interval (that may last until 2 hours each day); and if they complete them before the leaving time, they will rest. During free time and sleep time, workers will also rest, but the rest efficiency will be halved if the worker has remaining tasks.

Tasks arrive at the start of a work day and are distributed among the workers following a normal distribution with parameters $\mathcal{N}(20, 6)$. All tasks have a duration of 25 minutes, and the number of tasks that are commissioned to an agent along with his average number of daily tasks are used to calculate its event stress for that working day.

Once the worker arrives to work, time pressure is calculated every minute taking into account the remaining time to the leave hour and the required time of pending tasks. Lastly, effective fatigue is calculated every minute of the day, as it may decrease also when the user is resting at home. It is worth to analyse in detail the parameters that affect effective fatigue:

Temperature and humidity. In order to provide the simulation with a greater reliability, the system takes the values of temperature and humidity from an open weather API[1] for each hour of the next 10 days in the wanted city. In this case, the data has been obtained for the city of Madrid in the days between 2nd and 12th July, and saved in a csv that is loaded with Pandas library as a dataframe. Then, for each hour of each day, the temperature and humidity values are obtained and the PMF is calculated. In case that wet-bulb globe temperature is between the optimum values, the stressor will be positive and the effective fatigue will decrease; but if it is not, the stressor will be negative and the effective fatigue will increase following the equation described in Table 3.1.

[1]Weather API: https://www.apixu.com

Noise. The noise has been modeled with a random distribution following values extracted from a noise dataset generated by The National Institute for Occupational Safety and Health (NIOSH)[2], that provides several noise levels in different office contexts. The noise level increases or decreases between these levels depending on the number of workers and the average stress level. When the noise level is optimum, the effective fatigue decreases gradually; however, when it is high, effective fatigue increases as described in Table 3.1.

Luminosity. In the same way that noise, luminosity has been modeled with a random distribution following values obtained from a dataset. In this case, the dataset is the Multicom Domus Dataset [97], that provides luminosity values captured from two sensors. When the luminosity level is optimum, the effective fatigue decreases gradually; however, when it is high, effective fatigue increases as described in Table 3.1.

Email reception. The emails read in a day follow a normal distribution with parameters $\mathcal{N}(12.54, 8.02)$, proposed by Kostadin Kushlev and Elizabeth W. Dunn [87]. The email reception is distributed along the day for each worker, and when the worker receives an email, he spent a time (in minutes) reading it without doing any tasks. This time has been calculated in function of the time in reading the email (that uses to be about 25 seconds), the time in answering the email and the time in getting back to the task; resulting in a normal distribution with parameters $\mathcal{N}(3, 0.5)$. In addition, the email reception has a negative effect on workers' stress, as it increases effective fatigue as shown in Table 3.1.

Overtime hours. Working overtime hours will negatively affect stress as mentioned in Chapter 3, increasing effective fatigue following the equation of Table 3.1. The worker will work a maximum of 2 daily overtime hours if he has uncompleted tasks.

Rest time. Finally, the time that the worker is not working, decreases the effective fatigue following the equation of Table 3.1. However, if the worker has remaining tasks, the decrease of effective fatigue while the user is resting will be halved.

The above parameters as well as its contribution to workers' mood have been estimated with the purpose of making the simulation as reliable as possible. The effective fatigue is calculated every minute, and once the three components of stress have been calculated, are then summed to get a value of the total stress for each worker. There are five states for the worker depending on his stress level: lame [0, 0.2), inattention [0.2, 0.4), optimal [0.4, 0.6), fatigue [0.6, 0.8) and anxiety [0.8, 1]. Finally, the stress value is used for calculating

[2]Noise dataset: https://www.cdc.gov/niosh/data/datasets/rd-1005-2014-0/default.html

worker's productivity, as shown in Equation 3.7. Once the simulation parameters have been described, the privacy policies to evaluate will be analysed. The user may configure all these parameters by mean of the configuration files.

5.1.1 Privacy policies analysis

The collectors submodules of the simulation system architecture (*sensor collector*, *worker collector* and *time collector*) generate several results that enable the representation of stress and productivity levels depending on the ambient and work conditions.

Firstly, the result of each one of the strategies will be presented, and then, all of them will be compared in order to extract the conclusions.

5.1.1.1 Baseline

Baseline has been the first policy to analyse. The simulation has been performed for 10 days, with all the automation features disabled. Figure 5.1 shows an histogram with the workers grouped by their final stress level.

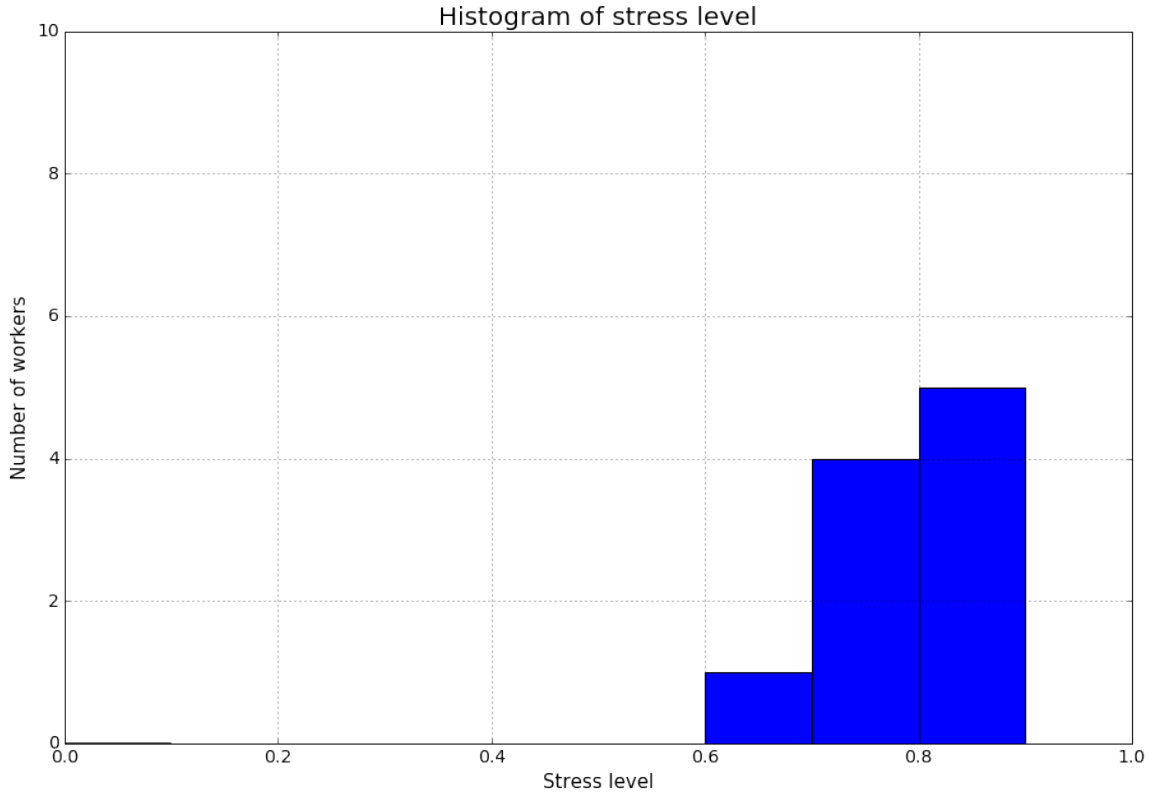

Figure 5.1: Baseline policy stress histogram

As shown in the figure, the stress level is quite high, as five workers reach the anxiety level (more than 0.8), and the rest feel fatigue. None of them has finished the 10 days in an optimal stress level, and this has a great impact on productivity, as shown in Figure 5.2. This figure represents the average productivity of all workers along with their stress for each hour of the 10 days.

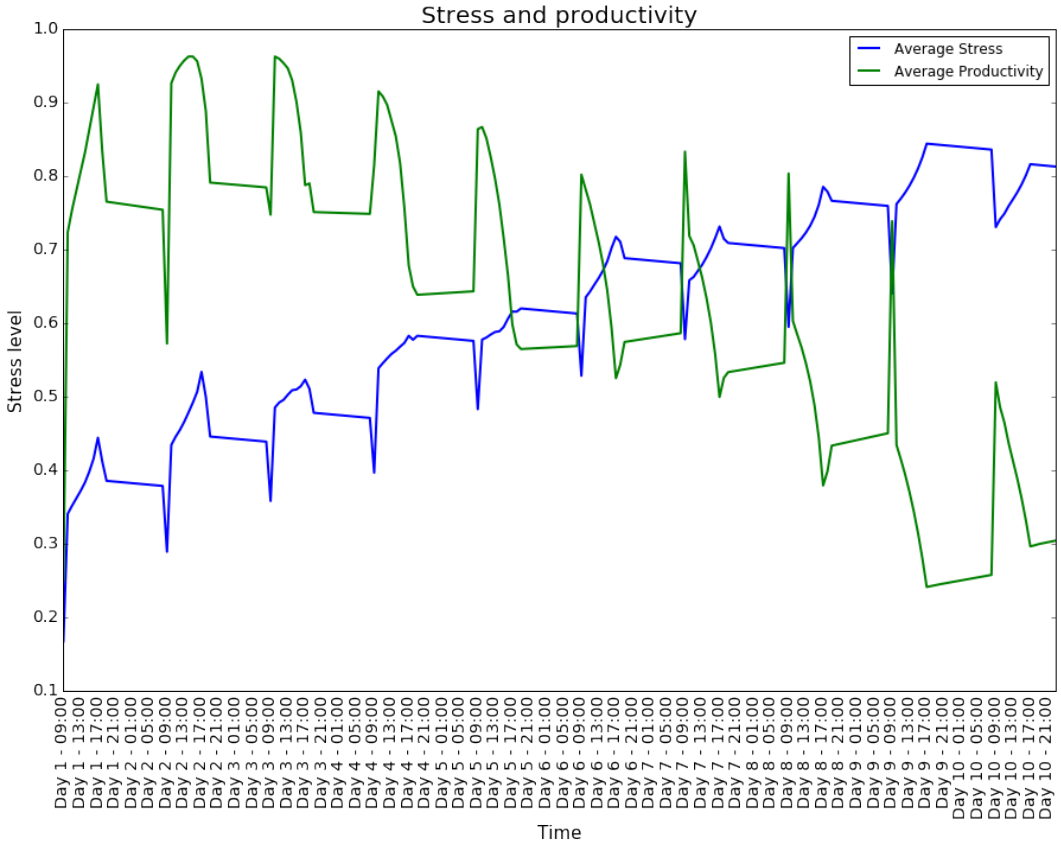

Figure 5.2: Baseline policy stress and productivity

As can be noted, the productivity reaches its pikes when the stress level is in the optimal interval, and when stress starts to rise, productivity dramatically falls until its minimum value. With regards to the stress behaviour, every day it has a pike during work hours and slightly decreases during rest hours. However, the rest time is not sufficient and stress level significantly increases every day.

In order to find the cause of the high stress level, Figure 5.3 shows the stress along with its three components: event stress, time pressure and effective fatigue.

The figure shows that event stress remains around the optimum value, so the amount of tasks is not a problem. However, the effective fatigue dramatically increases from the third day, reaching its maximum level at day 8. Since the huge rise of effective fatigue, time

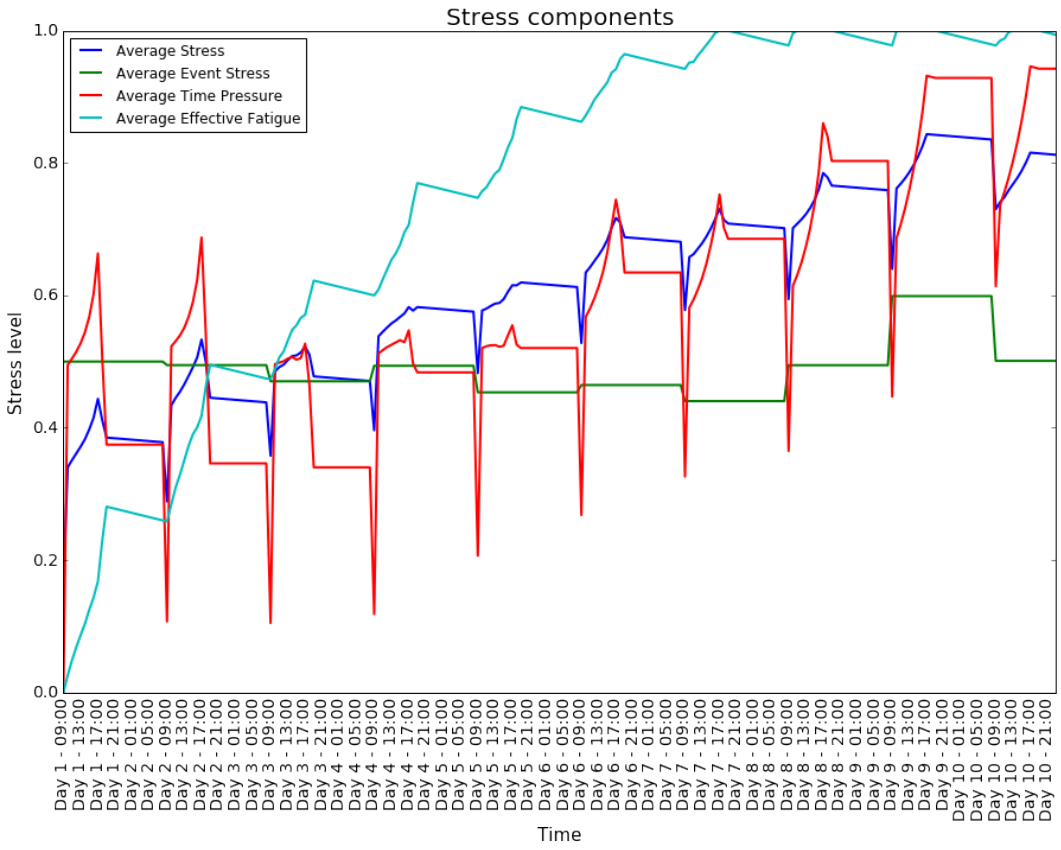

Figure 5.3: Baseline policy stress componentss

pressure also rises, as the fall in the productivity causes tasks to accumulate and workers become overwhelmed. In fact, comparing Figure 5.2 and Figure 5.3 can be seen that time pressure starts to increase when productivity falls under its medium value.

So the main cause of the high level of stress and its consequent low productivity is the effective fatigue of employees. This fact indicates a poor ambient and work conditions, and in order to deep in the problem, the ambient components of effective fatigue are presented in Table 5.1.

	Min	**Max**	**Avg**
WBGT	14 °C	30 °C	22 °C
Noise	45 dB	75 dB	61 dB
Luminosity	483 lux	539 lux	513 lux

Table 5.1: Ambient components of effective fatigue

Although the average of these values is in the optimum level, can be noted that wet-bulb globe temperature limit values are quite far from the optimum. This fact also happens at noise level, that has a maximum of 75 dB and an average value that is near the limit of 65 dB. The exposure of workers to high levels of noise and too low or high values of temperature for a long time may be one of the causes of the extremely high effective fatigue. In addition, Figure 5.4 shows that the higher temperature values coincides with the work hours.

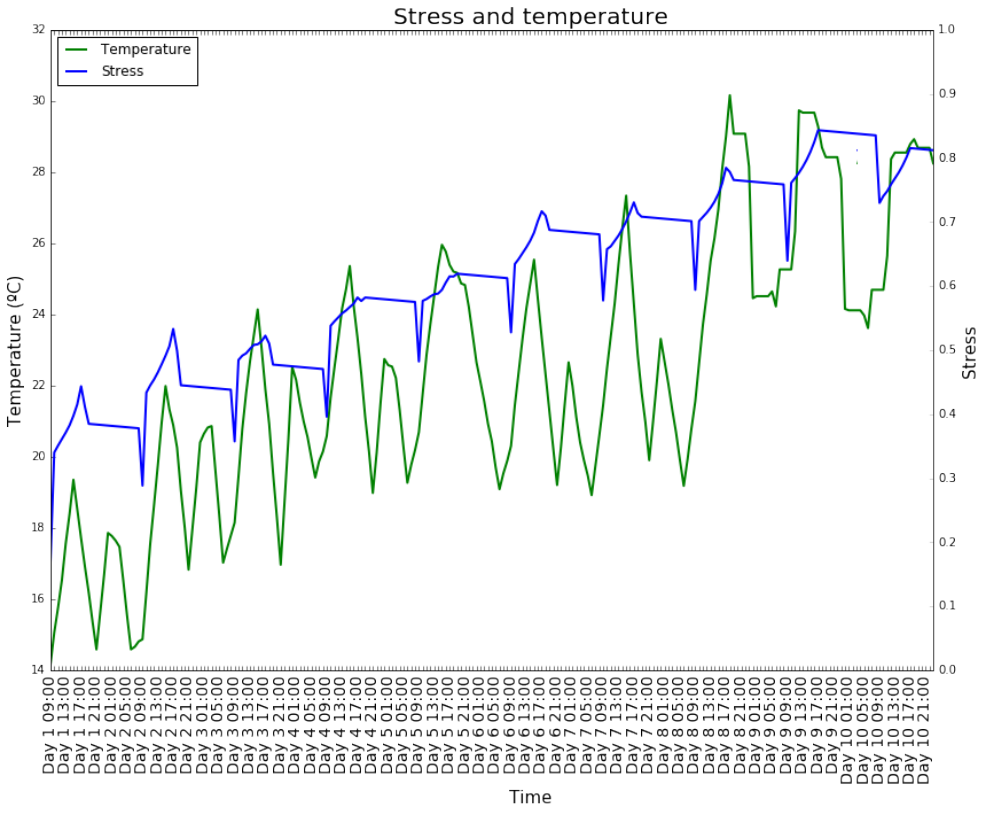

Figure 5.4: Baseline policy stress and temperature

Focusing on these results, temperature and noise seem to be two of the causes to effective fatigue. However, other factors that also contribute must be analysed.

	Max	**Avg**
Overtime hours	20	8
Emails read (in a day)	34	11
Pending tasks	118	21

Table 5.2: Baseline policy workload components of effective fatigue

Figure 5.2 shows some work conditions that also influence effective fatigue. As can be seen, the number of pending tasks is extremely high, which explains the time pressure rise commented before. In addition, the maximum level of overtime hours and emails is also too high. Finally, general results of the experiment related to stress level, productivity and completed tasks are presented in Table 5.3.

	Min	**Max**	**Average**
Completed tasks (in a day)	14	23	19
Productivity	0.11	0.99	0.62
Stress	0.17	0.92	0.61
Completed tasks (total)	-	-	2586
Productivity (last day)	0.25	0.68	0.36
Stress (last day)	0.67	0.83	0.78

Table 5.3: Baseline policy ambient components of effective fatigue

Once studied in detail the baseline policy, the ambient automation strategy analysis will be presented.

5.1.1.2 Ambient Automation

In this simulation, the control of ambient conditions will be automated. The user can easily configure this by mean of the automation configuration settings, that is shown in Listing 5.1. He has only to change to True the variables related to ambient conditions.

Listing 5.1: Automation configuration

```
automate_temperature = True
automate_noise = True
automate_luminosity = True
automate_emails = False
automate_tasks = False
```

Once the parameters have been changed, the simulation can be run. Again, in this case it has been run for 10 days, with 10 agents, and with the ambient conditions control

automation enabled. With this policy, ambient conditions will always be in the optimum value. However, the workload automation remains disabled, so only tasks related to adjust the ambient conditions (such as turning the lights or the HVAC system on or off) will be automated. Figure 5.5 shows an histogram with the workers grouped by their final stress level.

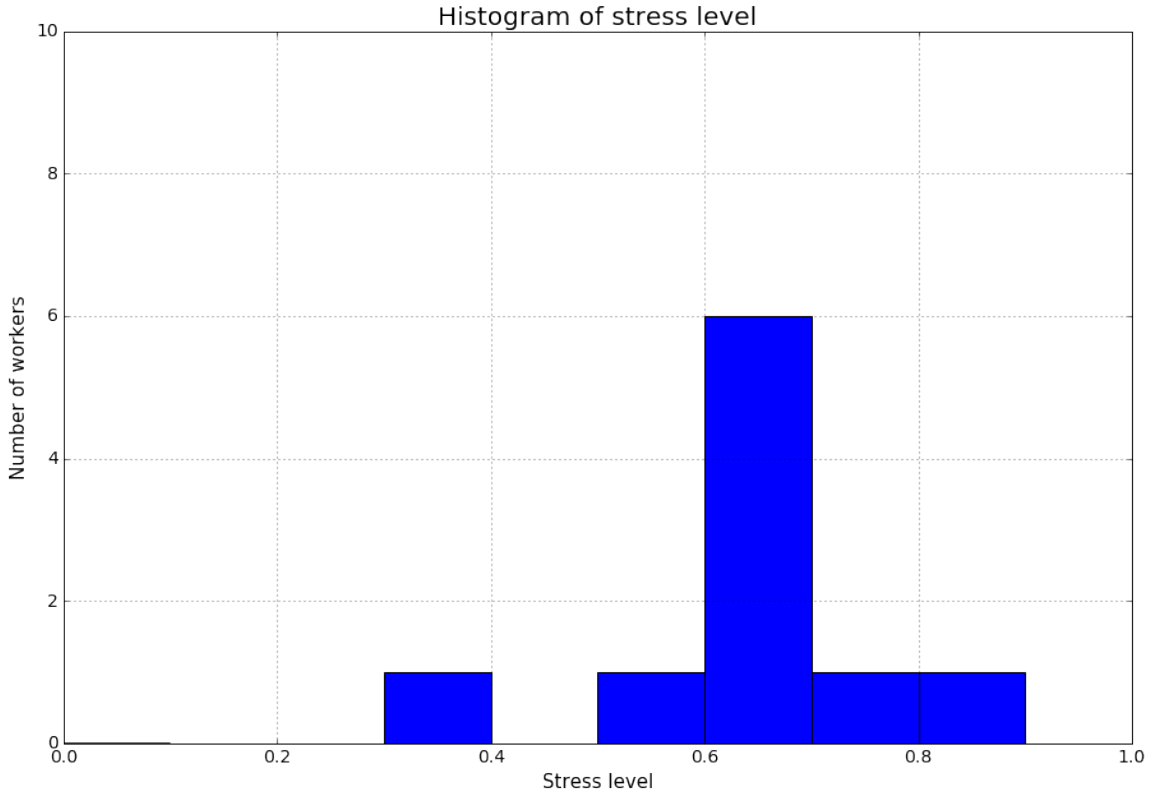

Figure 5.5: Ambient automation policy stress histogram

As shown in the figure, the stress level is now lower that in the first case, as only one worker reaches the anxiety level. However the the majority of the rest feel fatigue, and only one has an optimal stress level. With regards to the improvement of this policy to the productivity, analysing the results in Figure 5.6 can be seen that it has significantly improved.

Paying attention to the chart, can be noted that the productivity reaches its pikes when the stress level is in the optimal interval, and in this case that happens frequently. However, as shown in the figure, productivity also falls dramatically when the stress level is very low. With regards to the stress behaviour, every day it has a pike during work hours and slightly decreases during rest hours. However, the rise during work hours is not as significant as in the baseline policy, and even there are days that remains steady.

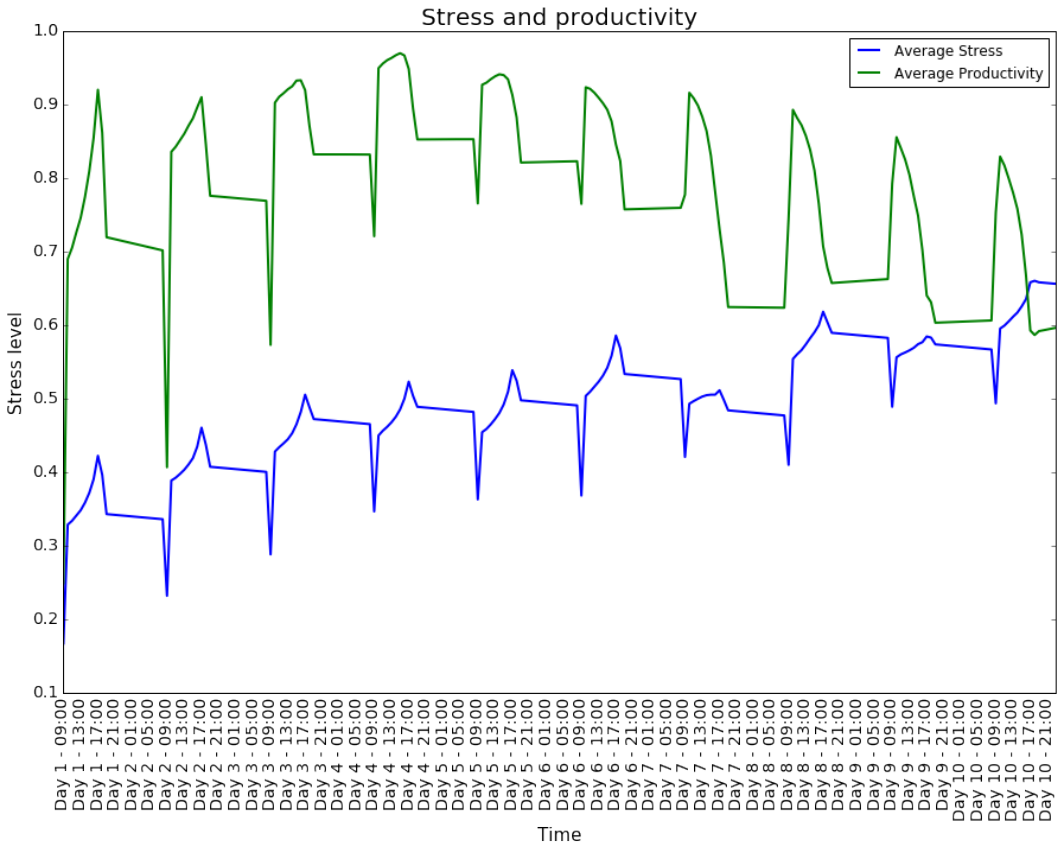

Figure 5.6: Ambient automation policy stress and productivity

In order to appreciate more in detail the effects of this policy, Figure 5.7 shows the stress along with its three components: event stress, time pressure and effective fatigue. In this way, changes in effective fatigue, that is the most affected component by ambient conditions, can be measured.

The figure shows that event stress remains around the optimum value, in the same way that in the first policy. However, the effective fatigue rises slower than in the baseline policy, ought to the lack of negative ambient stressor. In addition, time pressure also remains more stable, although has peaks at the start of the working day. The dramatically fall that suffers at the start of each day is ought to the increment in remaining work time at the arriving time, although it recovers when new tasks are assigned.

The study of this chart confirms that the policy of automating ambient conditions is useful. However, the tendency of the effective fatigue is to continue rising, so even though in this case this rise is slighter, if the simulation lasts a few more days, probably stress levels will recover.

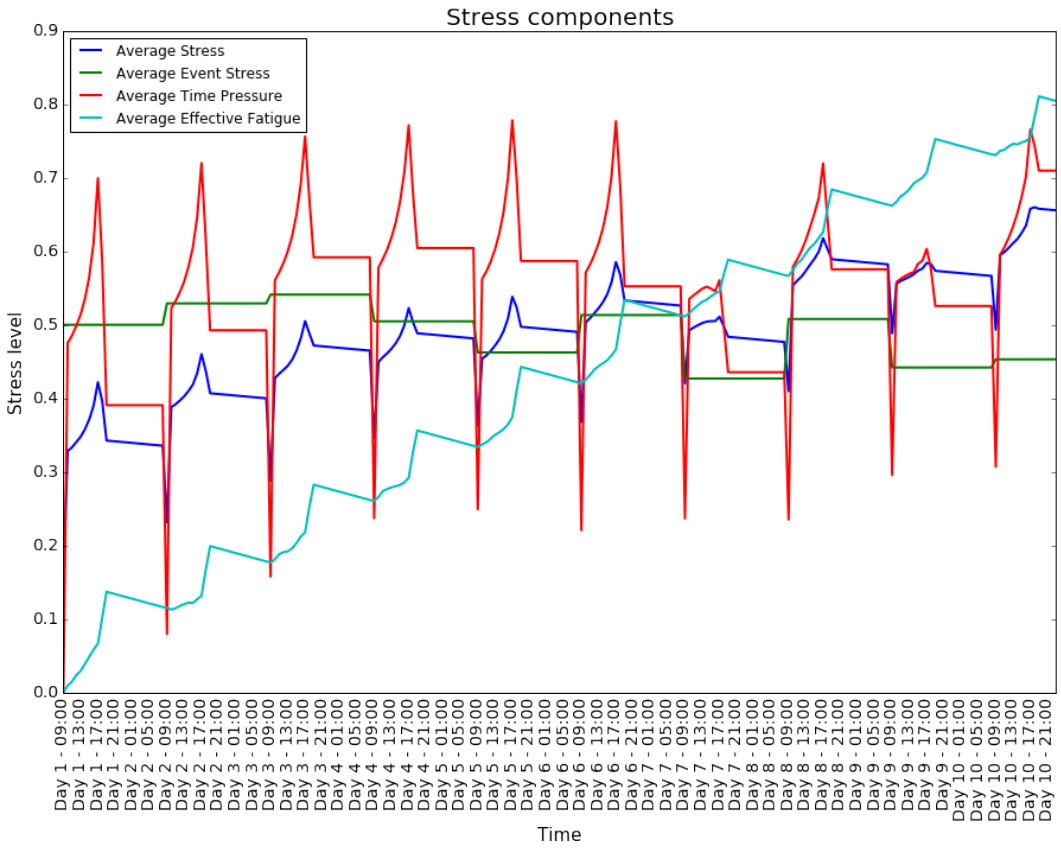

Figure 5.7: Ambient automation policy stress components

As the ambient conditions are being controlled automatically; this fact indicates that work conditions are not ideal, so they are presented in Table 5.4 in order to analyse them.

	Max	**Avg**
Overtime hours	20	9
Emails read (in a day)	41	11
Pending tasks	70	15

Table 5.4: Ambient automation policy workload components of effective fatigue

The pending tasks have significantly decreased with respect to previous strategy, ought to the increment in the productivity. However, it is still quite high, and paying attention to the table is easy to find a possible cause: the number of emails read continues being too big. This causes distractions in the worker, who spends too time reading emails and is not able to finish his tasks, so pending tasks rise, overtime hours rise, and consequently, stress

level rises and productivity falls. Finally, general results of the experiment related to stress level, productivity and completed tasks are presented in Table 5.5.

	Min	**Max**	**Average**
Completed tasks (in a day)	16	23	18
Productivity	0.21	0.99	0.77
Stress	0.16	0.85	0.49
Completed tasks (total)	-	-	3240
Productivity (last day)	0.32	0.88	0.72
Stress (last day)	0.33	0.80	0.62

Table 5.5: Ambient automation policy ambient components of effective fatigue

The table shows that all parameters have significantly improved with respect to the previous policy, demonstrating that it is worth to implement ambient conditions control automation in the office. The next step for the manager of the company is to simulate the workload automation policy.

5.1.1.3 Workload Automation

Finally, the workload automation strategy is going to be evaluated. The user has to change the corresponding values in the automation settings, and then, the simulation can be run. In this case, some tasks related to work will be automated. Thanks to these automations, the workers will be around a 10% more efficient. In addition, by automating the inbox email, the emails read has been reduced to a normal distribution with parameters $\mathcal{N}(4.70, 4.1)$ [87].

These parameters can be configured, as well as the rest of contributors to stress in the model settings file, in a similar way to the automation settings file. In addition, the email reception distribution can be configured in the email settings configuration file.

Once the automation has been configured, the simulation can be run. Figure 5.8 shows the histogram with the workers grouped by their final stress level.

As shown in the figure, the stress level has significantly fallen. Now most employees are in the optimum level, with a stress level between 0.4 and 0.6, and none of them reach the anxiety level, or even the fatigue level. However there are three workers who are in the

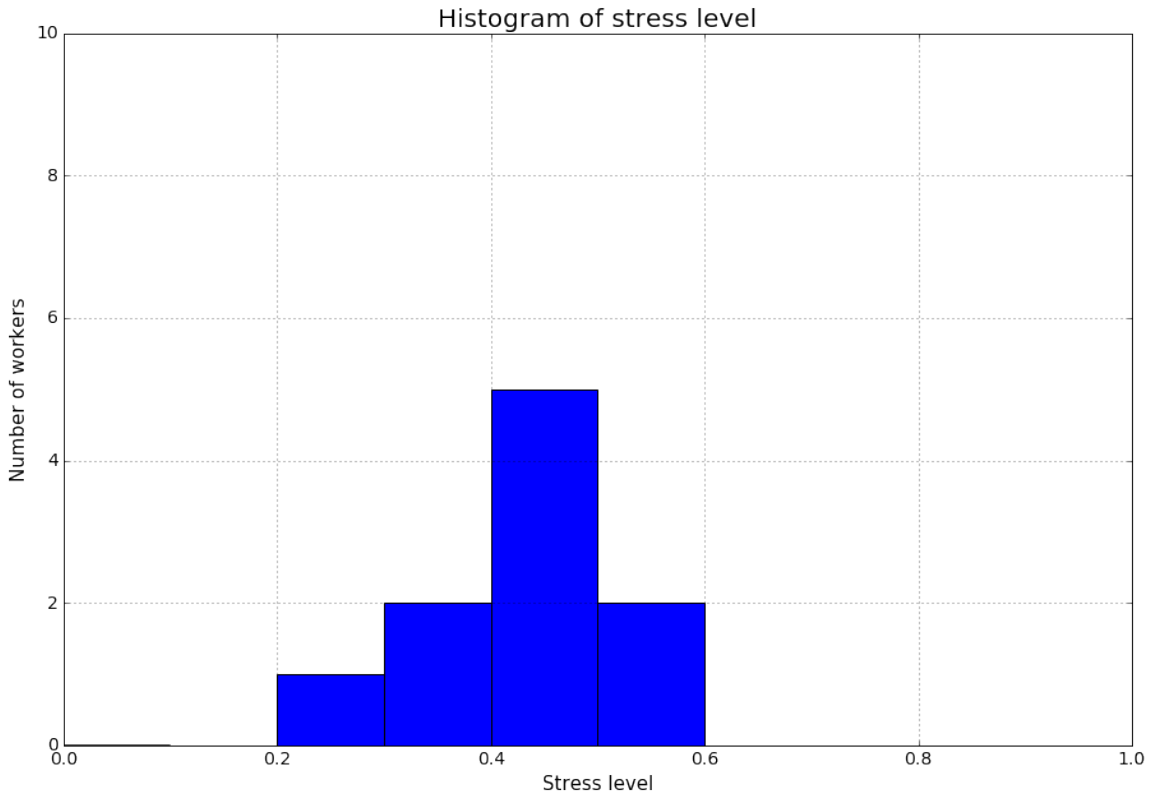

Figure 5.8: Workload automation policy stress histogram

inattention interval, and this also translates in a decrease of the productivity. Analysing the results in Figure 5.9 can be seen that although the level remains more or less in the same value that in the previous strategy, the stability is greater. This time, the stress level remains steady and there is not a reason to think that it will increase in the future, so the productivity will also remain in its current level.

In addition, can be noted that in worktime hours the productivity reaches its optimum level with values over 0.8, and that when it falls is when the user is not at work. The productivity reaches its pikes when the stress level is in the optimal interval, but falls when the stress level is too low.

The effects of this policy on each component of stress can be appreciated in Figure 5.10, that shows the stress along with its three components. The figure shows that event stress remains around the optimum value, in the same way that in the previous policies. However, the effective fatigue rises very softly and its value is much lower than in the previous experiments. In addition, time pressure also remains yet more stable, being in its optimum value during the work time.

The study of this chart confirms that the policy of workload automation is very useful

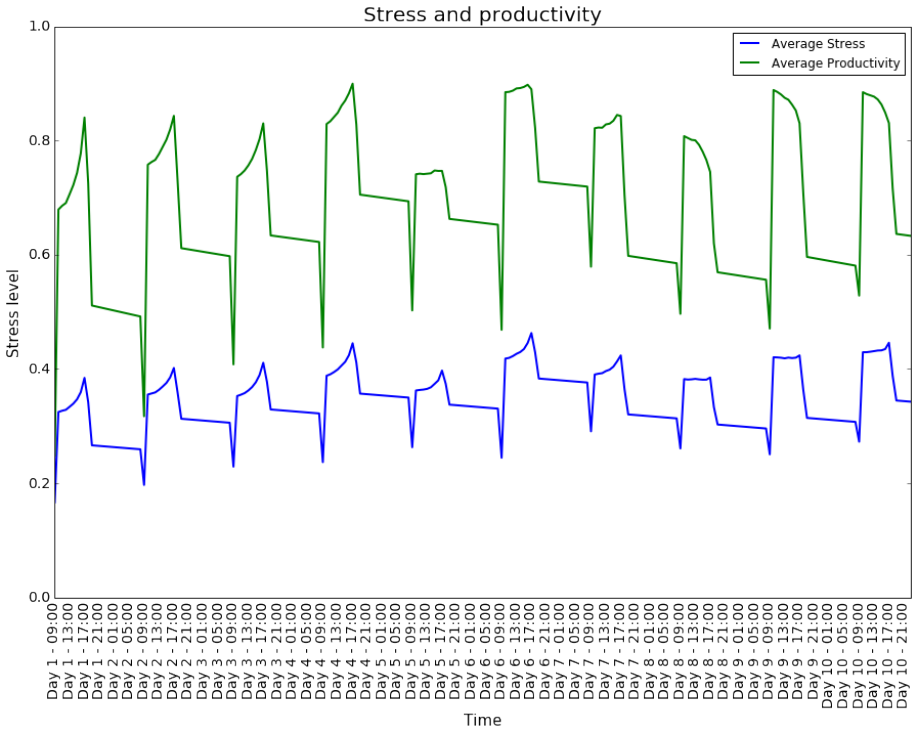

Figure 5.9: Workload automation policy stress and productivity

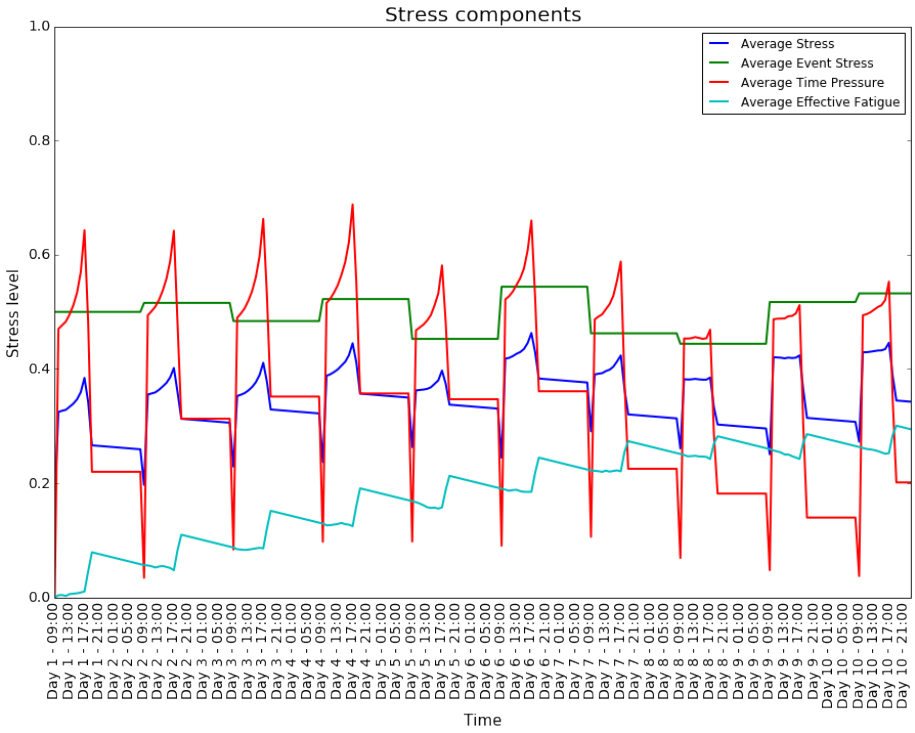

Figure 5.10: Workload automation policy stress components

for reducing stress, as all the elements that contribute to its increasing have been controlled. Ambient conditions in the previous policy. Work conditions are presented in Table 5.6 in order to analyse them.

	Max	**Avg**
Overtime hours	20	9
Emails read (in a day)	14	4
Pending tasks	41	7

Table 5.6: Workload automation policy workload components of effective fatigue

The pending tasks have significantly decreased with respect to previous strategy, ought to the increment in the productivity and in the worker efficiency. In addition, the number of emails read has fallen, so employees distractions have also decreased. Overtime hours, however, does not have any significant change with respect to previous policy. Finally, general results of the experiment related to stress level, productivity and completed tasks are presented in Table 5.7.

	Min	**Max**	**Average**
Completed tasks (in a day)	16	26	22
Productivity	0.12	0.99	0.68
Stress	0.09	0.60	0.34
Completed tasks (total)	-	-	3383
Productivity (last day)	0.41	0.99	0.86
Stress (last day)	0.23	0.56	0.43

Table 5.7: Workload automation policy ambient components of effective fatigue

The table shows that productivity mean has been softly reduced with respect to the previous strategy, ought to the too low stress levels. However, paying attention to the last day, the parameters have improved significantly, and both productivity and stress have their best values of the three policies. After analysing each policy separately, in the following section they will be compared.

5.1.2 Results

In this section, the three policies will be compared in order to facilitate the drawn of conclusions. The first component to compare is the average stress over the time, shown in Figure 5.11.

The stress level is significantly lower in the workload automation policy, followed by the ambient automation policy. In addition, while with baseline and ambient automation strategies the stress level rises every day, with the workload automation policy, it remains stable between 0.2 and 0.4. However, these values are slight low, as values under 0.4 have negative influence at workers' productivity.

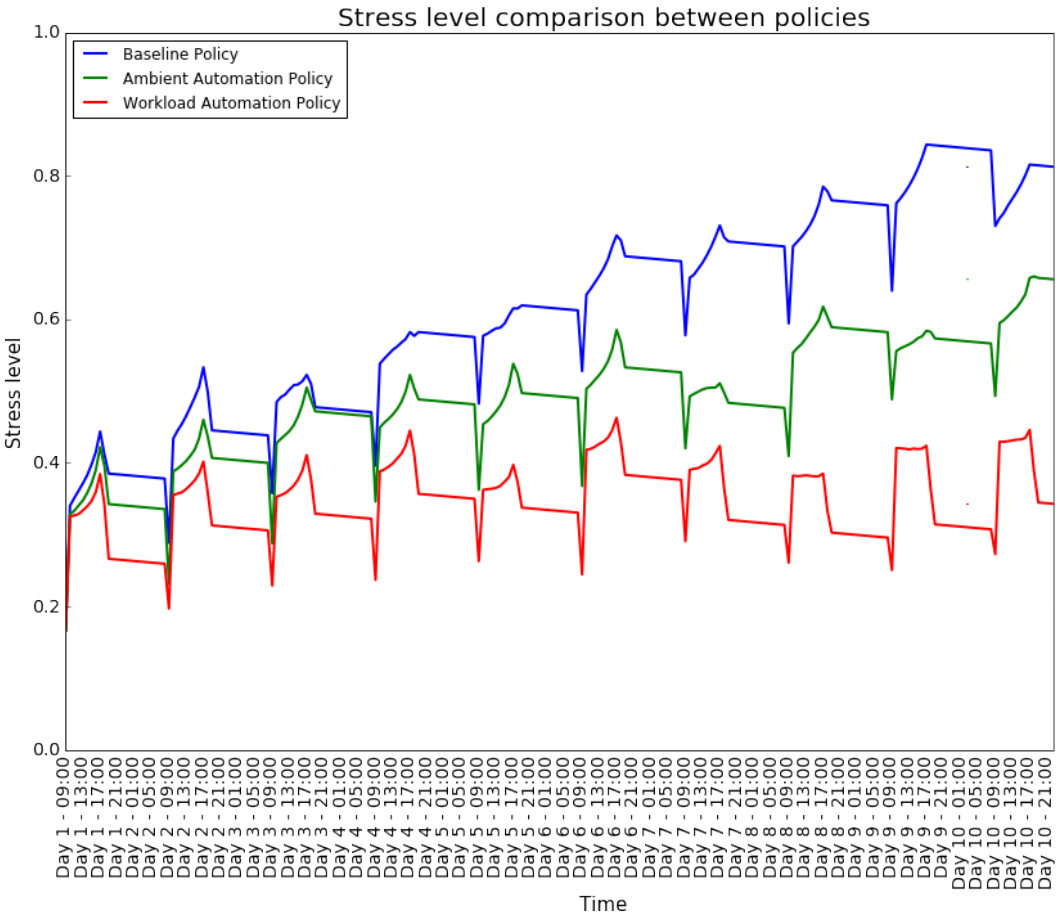

Figure 5.11: Average stress evolution of the three policies

With the purpose of analysing more in detail the effect that each policy has on stress, the Figure 5.12 shows a comparison between the average value of the different components of stress for each policy. As depicted in the figure, while event stress keeps stable for the three policies, the other two components suffer a great fall. The ambient automation policy

has a great impact on effective fatigue, and time pressure also experiments a soft fall ought to the decrease in pending tasks number. However, the impact of workload automation is significant for both policies, as it directly affects to the worker efficiency (preventing tasks accumulation) and emails read (preventing distractions).

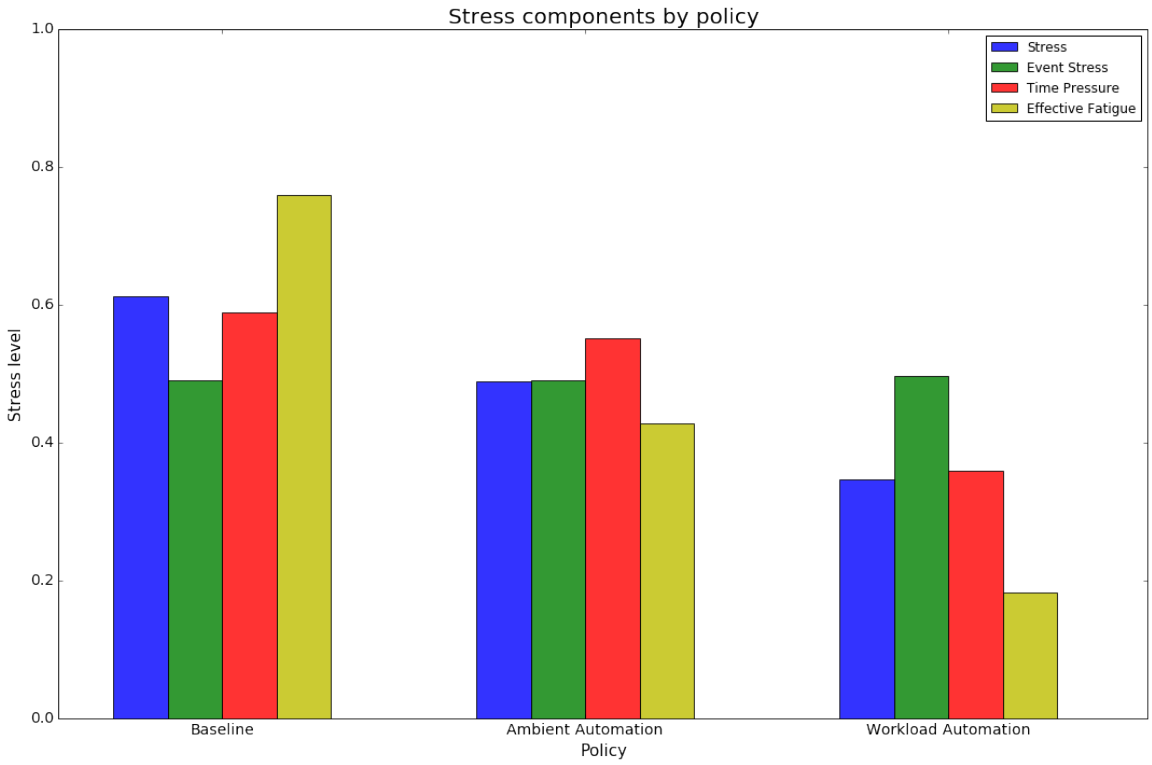

Figure 5.12: Average stress components levels

These figures demonstrate that from the point of view of reducing stress, the implementation of an automation platform in the office is worth. And the automation of both ambient and workload conditions controlling is the best policy to follow. However, before making a decision, the company manager must also study the evolution of workers productivity for the different strategies. For this reason, some relevant extracted parameters related to productivity are shown in Table 5.8.

The table shows four values of productivity: average, minimum, maximum and the average value in the last day of the simulation; four about completed tasks: minimum, maximum, average and total completed tasks by all workers during the simulation; and two values of pending tasks: maximum and average.

As shown, most parameters significantly improve with the application of automation policies. However, the average productivity decreases for the workload policy ought to the low stress level acquired.

	Productivity				*Completed tasks*				*Pending tasks*	
Policy	*Last*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Total*	*Max*	*Avg*
Baseline	0.36	0.11	0.99	0.62	14	23	19	2586	118	21
Ambient automation	0.72	0.21	0.99	0.77	16	23	21	3240	70	15
Workload automation	0.86	0.12	0.99	0.68	16	26	22	3383	41	7

Table 5.8: Comparison between policies for productivity related parameters

The obtained results open a wide range of possibilities for the company manager. It seems obvious that the implementation of at least one of the automation policies would be a great option for the company; however, the manager may wonder if would be better the ambient automation or the workload automation. Analysing the results, he could think in two options: to implement the ambient automation and try to reduce the number of tasks in order to prevent a raise in the effective fatigue level caused by overtime hours; or to implement the workload automation and to increment the number of tasks, increasing in this way the event stress of workers in order to prevent extremely low stress levels.

One of the key features of the proposed simulation system, is that the company manager could easily run again a simulation with the two above mentioned options, only by changing the configuration settings. In this way, he would be able to study the results of each of them and choose one with more security, or even to change more parameters until he gives with the optimal policy.

5.2 Emotion recognition

After evaluating by mean of the simulation system several options for the implementation of the automation platform in the smart office, the company manager has decided to implement the workload automation policy. With this policy, not only work related conditions control will be automated, but also ambient conditions.

Specifically, the manager has opted for implementing a system that detects temperature, humidity and noise as ambient conditions. Luminosity was discarded given its low impact shown in the simulation. In addition, the implemented system will be able to detect workers' emotion from speech and face recognition. Finally, several Internet services such as Github, Trello, Google Calendar; and devices such as a smart light, a connected door or a smart

tv will also be integrated. In order to test all these devices before placing them in the real scenario, the manager has used the visualization module of the simulation system proposed, that is shown in the Figure 5.13.

Figure 5.13: Smart office visualization

Once defined the requirements of the system, the deployment of the platform can start. The installation of EWE Tasker is pretty straightforward, as it can be carried out by mean of Docker[3], and the emotion recogniser block consists of a Python program that can be run in any computer. The integration with Internet services and ambient sensors is explained in detail in previous work [92], so this scenario will focus on the integration with the emotion recogniser.

A new channel must be defined in EWE Tasker Core for representing the emotion recogniser. The channel only has one event: emotion detected, that is create by the event discoverer of the speech of face analyser, the one who has detected the emotion. A representation of this channel written in Notation 3 is given at Listing 5.2.

[3]https://www.docker.com

Listing 5.2: Emotion detector channel

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ewe: <http://gsi.dit.upm.es/ontologies/ewe/ns/> .
@prefix ewe-emodet: <http://gsi.dit.upm.es/ontologies/ewe-emodet/ns/> .

ewe-emodet:EmotionDetector a owl:Class ;
    rdfs:label "Emotion detector"@en ;
    rdfs:comment "This channel represents a emotion detector."@en ;
    rdfs:subClassOf ewe:Channel .

ewe-emodet:EmotionDetected a owl:Class ;
    rdfs:label "Emotion Detected"@en ;
    rdfs:comment "This Trigger fires every time your emotion detector
        detects an emotion."@en ;
    rdfs:subclassOf ewe:Event ;
    rdfs:domain ewe-emodet:EmotionDetector .
```

Once the channel has been created and the system has been integrated, the task automation process can start and workers are able to create their own automation rules by using the EWE Tasker application web or its mobile application. The platform enables the creation of countless automation rules, in the following sections some applications that have special interest for both the worker and the manager will be described .

5.2.1 Emotion regulation

A possible application of the system is trying to affect the worker when a negative emotion has been detected in order to decrease its intensity or even change it by a positive emotion. Some examples of rules with this application are the following:

(a) *If I'm angry, then play a relaxing song.* When an emotion detected event is triggered and the emotion detected is anger, a relaxing song will be played in the music player of the worker, in order to try to change his mood.

(b) *If I'm sad, then tell me a joke.* When an emotion detected event is triggered and the emotion detected is sadness, his smart personal assistant (such as Google Home or Alexa) will tell him a joke, in order to try to make him laugh.

These rules are created by the own worker, that must have previously configured the external services that participate in the rules (music player and Google Home in this case). The emotion change application helps to improve the mood of the workers and to keep them happy and motivated.

Listing 5.3: "If I'm sad, then tell me a joke" example rule

```
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ewe: <http://gsi.dit.upm.es/ontologies/ewe/ns/#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ewe-detector: <http://gsi.dit.upm.es/ontologies/ewe-emodet/> .
@prefix onyx: <http://www.gsi.dit.upm.es/ontologies/onyx/> .
@prefix wn-affect: <http://gsi.dit.upm.es/ontologies/wnaffect/ns#> .
@prefix ewe-gh: <http://gsi.dit.upm.es/ontologies/ewe-gh/ns/#> .

{
    ?event rdf:type ewe-emodet:EmotionDetected.
    ?event ewe-emodet:hasDetected ?emotion.
    ?emotion onyx:hasEmotionCategory wn-affect:Sadness .
    ?emotion!onyx:hasEmotionIntensity math:greaterThan 0.95.
}
=>
{
    ewe-gh:GoogleHome rdf:type ewe-gh:TellJoke.
}.
```

5.2.2 Emotion log

From the point of view of the company manager results very interesting to be able to know the mood of the workers. This collected information would be anonymously but would provide useful information for the manager, that may devise strategies for motivating them if their mood state is negative. Some examples of rules with this application are:

(a) *If many workers are sad, then send me an email.* When several emotion detected events are triggered in a short interval of time and the emotion detected is sadness, the manager may want to know it as soon as possible, because there may be any problem.

(b) *If many workers are angry, then save current ambient condition information.* When several emotion detected events are triggered in a short interval of time and the

emotion detected is anger, may be as a consequence of bad ambient conditions, so save ambient information for being analysed later.

(c) *If any emotion is detected, then save it.* When a new emotion event is triggered, whichever is the detected emotion, save it. This enables the use of this information for devising motivation plans for the workers or group therapy.

These rules are defined by the manager, and all the data related to worker emotions would be completely anonymous. The emotion log provides countless applications that allow the manager to better know the needs of the workers.

5.2.3 Evaluation

Once the manager and the workers have configured their rules, the task automation is enabled. Consider an scenario where the above described rules have been configured. One day, one of the workers of the company has the facial expression shown in Figure 5.14. The face recogniser GUI shows two screens, one with the real captured image, and the other shows the real captured image with an overlayed emoji representing the detected emotion. This module detects that he is angry, and the corresponding event is generated by the event discoverer submodule and sent to the automation platform.

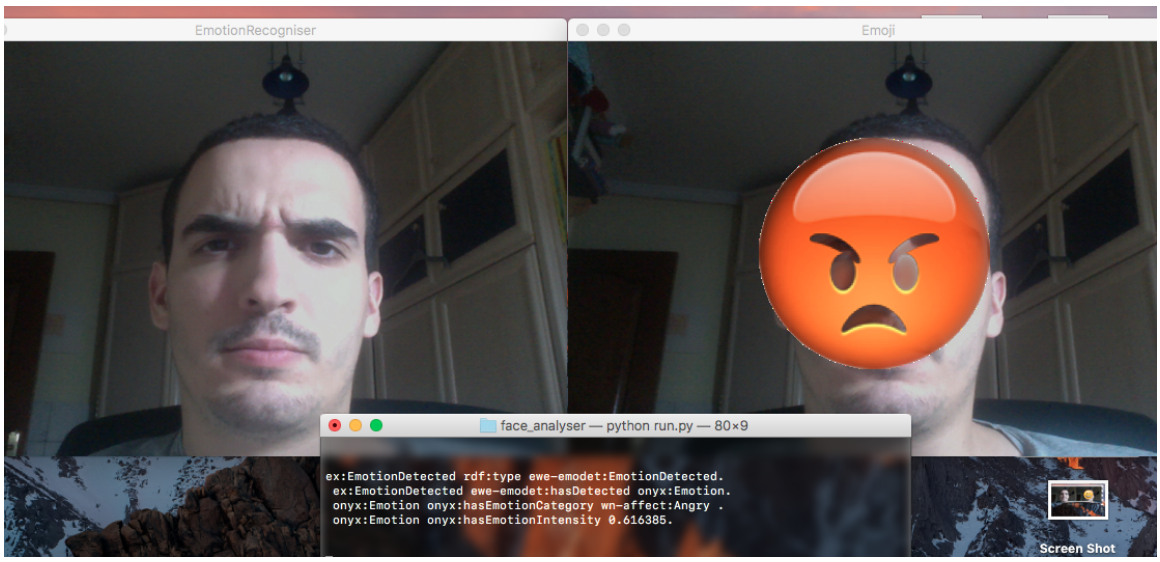

Figure 5.14: Anger emotion detection with the face analyser

The automation platform receives this event written in Notation3, as shown in the Figure, that contains the emotion detected and its intensity, and evaluates it along with the configured rules. As a result of the detection of anger event, two actions are triggered:

a relaxing song will be played, in order to change the worker's mood; and the emotion will be stored, in order to realize further analysis. The second rule does not impact directly on the worker, however, to play the relaxing song makes him feel more relaxed and his mood changes. In this way, the detection of a new emotion, represented in 5.15, confirms that the rule has successfully regulated the anger of the worker.

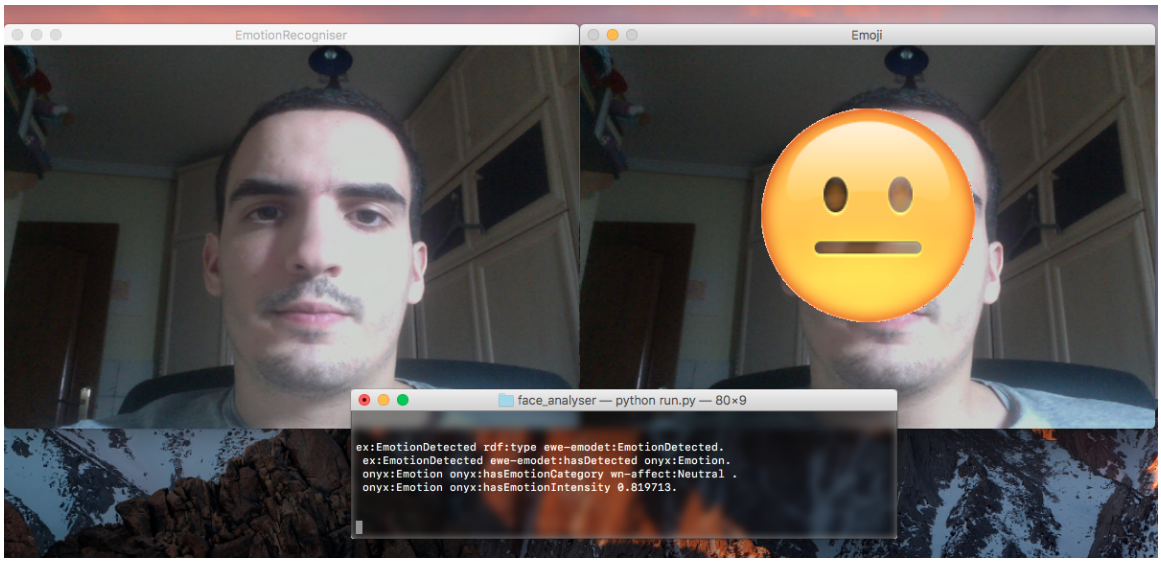

Figure 5.15: Neutral emotion detection with the face analyser

However, the next day, speech recogniser detects that several workers are angry in a short time interval, as shown in Figure 5.16. Both emotion detection events are sent to the automation platform, where they are evaluated.

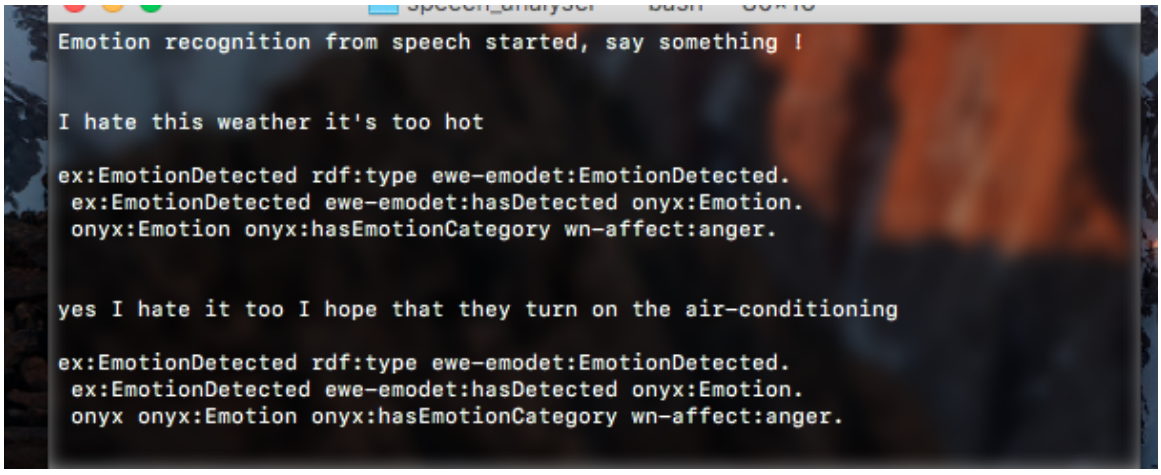

Figure 5.16: Anger emotion detection in two workers with the speech analyser

As result to the evaluation of the events, both rules triggered in the previous case are also triggered. Furthermore, a third rule is triggered, the one whose condition is that

several workers feel angry in a short time span, that stores ambient conditions in order to be analysed. The analysis results show that the air conditioning is not working correctly, so they fix it and the temperature falls, impacting workers' mood as shown in Figure 5.17.

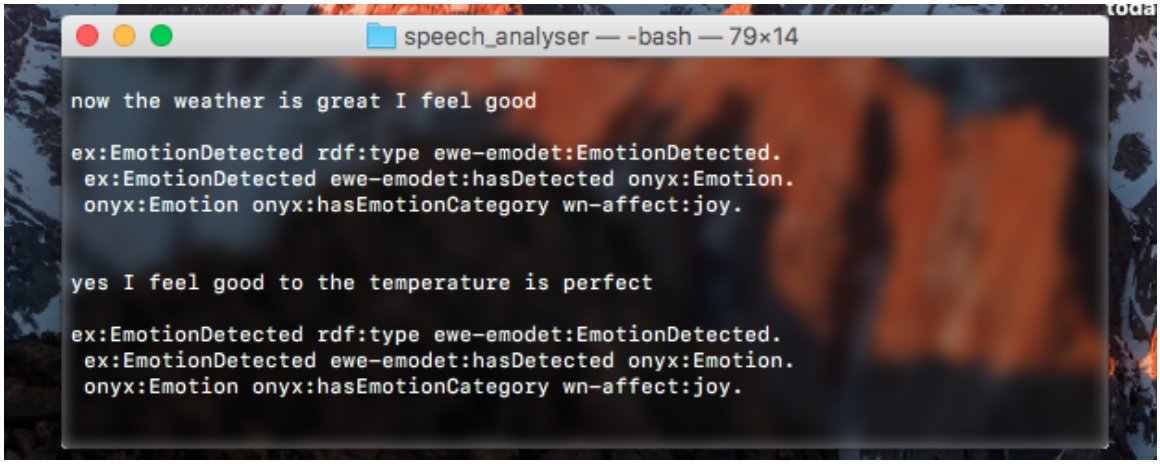

Figure 5.17: Joy emotion detection in two workers with the speech analyser

The next day, a worker feels sad, and the face analyser detects his emotion as shown in Figure 5.18. The generated sad emotion detection event is sent to EWE Tasker and evaluated, and the action of make Google Home assistant to tell a joke to the worker is triggered.

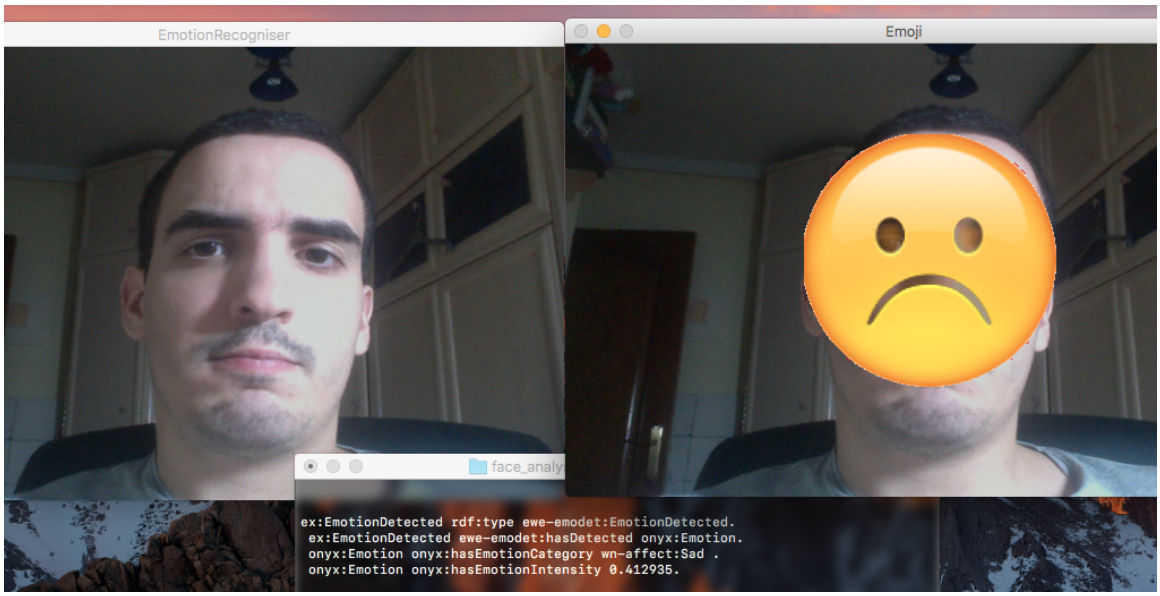

Figure 5.18: Sadness emotion detection with the face analyser

This action makes laugh to the worker, and regulates his emotion. In this way, his emotion changes from sadness to happiness, as shown in the Figure 5.19. It results also

interesting to evaluate the intensity of emotions. The Figure 5.19 shows a detection of happiness with an intensity of 0.61. However, his mood remains improving, and the next detected emotion is also happiness, but in this time the intensity has risen until reaching a value of 0.93, which represents a considerably high intensity for the emotion.

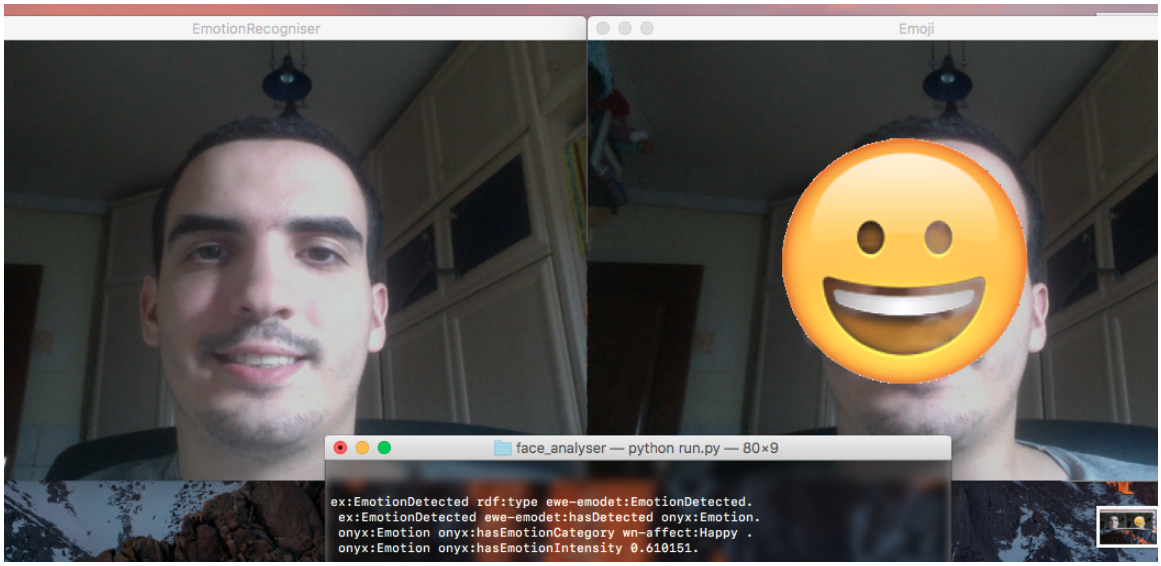

Figure 5.19: Moderate happiness emotion detection with the face analyser

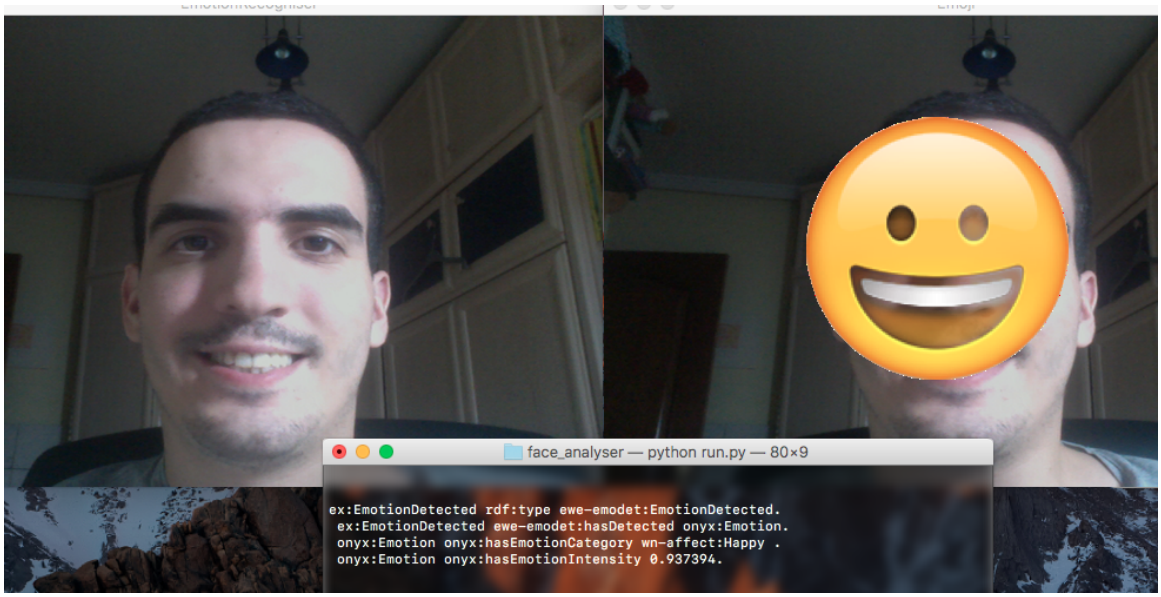

Figure 5.20: High happiness emotion detection with the face analyser

The above described are some of the applications of the system. However, as explained above, the proposed system provides functions for making straightforward the process of integrating new devices or services that extend the current applications.

Finally, several metrics have been evaluated in order to obtain an objective measure of the face recogniser module. The module has been tested with a dataset of 3589 images extracted from fer2013 [98]. The results show a precision of 0.559, a recall of 0.560 and a f-score of 0.555. These values are satisfactory, being comparable to the current state of the art. Other works such as Gera et. al. [99] report an accuracy of 54% combining audio and video, so our result of 55% using only video analysis can be considered a good value. Furthermore, analysing the confusion matrix represented in Table 5.9, can be noted that it has great values for some emotions as happiness, where it achieves a precision of 0.79.

	Anger	**Fear**	**Happiness**	**Sadness**	**Surprise**	**Neutral**
Anger	**244**	43	52	110	18	79
Fear	64	**165**	53	128	49	69
Happiness	41	16	**698**	51	21	52
Sadness	66	47	54	**296**	15	116
Surprise	19	40	40	19	**274**	24
Neutral	48	30	68	135	12	**333**

Table 5.9: Confusion matrix for the detected emotions from face analyser

To conclude, a briefly overview of the case study is given. The case study focuses on a company that aims to implement an intelligent environment at its office. With this purpose, the company manager uses the proposed agent-based simulation system for obtaining the best policies to follow in the implementation, taking into account emotional parameters of agents such as stress or fatigue in order to make the simulation as reliable as possible.

Once decided the requirements of the system and having simulated its behaviour, the company manager implements the proposed emotional context aware automation platform in order to benefit from the opportunities of emotion recognition. Results can be rapidly noted in the company, whose productivity increases as a consequence of the positive mood of the workers and the optimal ambient and work conditions.

CHAPTER 6

Conclusions and future work

This final chapter describes the conclusions of this master thesis, as well as possible lines of future work. The conclusions offer a brief summary of the work developed during this master thesis, focusing on the main results that have been obtained; while the future work describes some design and implementation ideas that would be interesting to implement for this master thesis.

6.1 Conclusions

In this master thesis, the development of an emotion aware automation platform has been proposed, resulting in a system that allows the user to configure his own automation rules depending on several factors and data coming from different sources, such as ambient conditions, Internet services or the detected emotion. With this purpose, the state of art related to emotions have been exhaustively studied, analysing the different ways of detecting them, the forms of representation and their most relevant aspects.

This project focuses on the smart office context, where automation rules can enable workers' emotion regulation, improving the work conditions, the workers' mood and consequently the productivity.

Two emotion detection modules have been developed, that detect emotions either from the facial expression or from the speech. The face recognition module has been developed with machine learning techniques, using a convolutional neural network architecture for designing a model that can predict six emotions: anger, sadness, happiness, surprise, fear and neutral. For the developing of this module, an exhaustive study of machine learning techniques has been done. With regards to the speech recognition module, it transcribes the voice to text, and extracts the emotion from it using Senpy tool.

Both modules have been fully integrated with EWE Tasker, a task automation platform developed on a previous work, that enables semantic event-driven task automation. With the purpose of taking advantage from the benefits of semantic technologies, emotions have been modeled using Onyx ontology, while automation rules are modeled using EWE. The application of semantic technologies provides the system with all the advantages of linked data, and facilitates the integration of new devices and services, which results crucial in an IoT system.

In addition, with the purpose of being able to test the system before the deployment in a real scenario and anticipate possible faults, an emotional agent-based social simulation system has been also developed. The system enables the simulation of a smart office, where ambient and work conditions influence workers' behaviour and impact on their stress and productivity. With this purpose, an emotional social model for stress has been implemented. This model takes into account several parameters related to ambient and work conditions such as task arriving, pending emails, temperature and noise to calculate the stress and productivity of the workers, and enables the realization of simulations for detecting the influence of these parameters at the productivity of the company. In addition, the system can be used for testing automation rules and automation policies before its implementation

in a real scenario.

Finally, the whole system has been evaluated by mean of its application to a case study where a company manager explores some of its functionalities. First, the simulation allows him to test several automation policies in order to find the optimal automation level that must implement at his company for obtaining the best results. Once analysed several policies, the manager chooses one, which is implemented in a real scenario, where the emotion detection modules as well as its integration with the automation platform is evaluated, demonstrating its good results and its utility to carry out emotion regulation. In addition, an objective evaluation by the use of several metrics demonstrates that the results obtained with the facial emotion detection are satisfactory, being comparable to the current state of the art

6.2 Achieved goals

This document started with a set of goals to be achieved with this master thesis. This section provides a summary of the final outcomes, in order to evaluate the fulfillment of that goals.

Development of an emotion aware automation platform. This was the main goal of this project, to develop a system that allows the user to automate tasks that have his emotion as a data source. The system has been successfully developed and applied to a real scenario.

Development of an emotional agent-based simulation system. The developed simulation system enables the test and evaluation of several automation policies in the smart office context, and provides a method for analysing the best policies to follow in the implementation of the automation platform to a real scenario.

Semantic modeling. The application of semantic technologies to the system significantly faclitates the integration of new components and services. Emotions as well as automation rules (including the elements that participate on them such as devices and services) have been modeled using semantic technologies in order to facilitate the integration with other systems.

Scalability and interoperability of developed modules. The easy integration with other systems is a key factor of the project, as IoT technologies are in continuous evolution. The target scenario of the system developed in this master thesis comprises

several actors (human and devices) that interact among them. In this system, all modules have been implemented using Python, guaranteeing its compatibility with most computer systems.

Real time emotion recognition. The modules developed in this thesis enable real time emotion recognition, being able to detect emotions at the same time that they are expressed.

Integration with other projects. The system proposed in this master thesis has been integrated with several projects in order to extend its capabilities. The integrated projects have been EWE Tasker, SOBA, RAMEN and Senpy.

6.3 Future work

There are many lines that can be followed to continue this work. The high scalability offered by the developed systems facilitates the extension of both the architecture and the developed tools with the purpose of giving a more solid solution to a wider range of scenarios.

Improve the simulation model. To extend the simulation model with new features such as the integration of the detected emotions in the real system would widely extend its applications, and would be a quantitative improvement in terms of reliability.

Implementation of emotion detection from voice features. Although the emotion detection from the transcribed text results a good approach, the implementation of emotion detection from voice features would mean a quality increase in the emotion detection, as it would not only take into account what is said, but how it is said.

Development of a GUI for the simulation system. A user interface where all parameters of the simulation could be configured would significantly increase the facility to use it, as it would not be necessary to manually change the configuration files.

Training of the face recognition model. The face recognition model could be trained with more datasets in order to increase its accuracy, and even it could be improved in order to get better precision.

Emotion detection from different sources. The integration of new systems of emotion recognition, such as the use of biometric sensors or pupil detection would increase the possibilities of the system, as new emotions and relevant data related to users could be detected.

Bibliography

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[3] J. C. Augusto, "Ambient intelligence: The confluence of ubiquitous/pervasive computing and artificial intelligence," *Intelligent Computing Everywhere*, pp. 213–234, 2007.

[4] K. FURDIK and G. LUKAC, "The Network Architecture Designed for an Adaptable IoT-based Smart Office Solution," *International Journal of* {...}, vol. 1, no. 6, pp. 216–224, 2013.

[5] S. Mennicken, J. Vermeulen, and E. M. Huang, "From today's augmented houses to tomorrow's smart homes," *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*, no. September, pp. 105–115, 2014.

[6] M. Coronado, C. A. Iglesias, and E. Serrano, "Modelling rules for automating the Evented WEb by semantic technologies," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7979–7990, 2015.

[7] S. Muñoz, A. F. Llamas, M. Coronado, and C. A. Iglesias, "Smart office automation based on semantic event-driven rules.," in *Intelligent Environments (Workshops)* (P. Novais and S. Konomi, eds.), vol. 21 of *Ambient Intelligence and Smart Environments*, pp. 33–42, IOS Press, 2016.

[8] B. E. Kok, K. A. Coffey, M. A. Cohn, L. I. Catalino, T. Vacharkulksemsuk, S. B. Algoe, M. Brantley, and B. L. Fredrickson, "How Positive Emotions Build Physical Health : Perceived Positive Social Connections Account for the Upward Spiral Between Positive Emotions and Vagal Tone," *Psychological science*, vol. 24, no. 7, pp. 1123–32, 2013.

[9] V. T. Nguyen, D. Longin, T. V. Ho, and B. Gaudou, *Integration of Emotion in Evacuation Simulation*, pp. 192–205. Cham: Springer International Publishing, 2014.

[10] M. A. Pervez, "Impact of emotions on employee's job performance: An evidence from organizations of Pakistan," *OIDA International Journal of Sustainable Development*, vol. 1, no. 5, pp. 11–16, 2010.

[11] H. Selye, *Stress without Distress*, pp. 137–146. Boston, MA: Springer US, 1976.

[12] W. Kirsten, "Health and productivity management in Europe," *International Journal of Workplace Health Management*, vol. 1, no. 2, pp. 136–144, 2008.

[13] R. Yerkes and J. Dodson, "The Relation of Strength of Stimulus to Rapidity of Habit-Formation," *Journal of Comparative Neurology and Psychology*, vol. 18, pp. 459–482, 1908.

[14] D. F. Dinges, S. Venkataraman, E. L. McGlinchey, and D. N. Metaxas, "Monitoring of facial stress during space flight: Optical computer recognition combining discriminative and generative methods," *Acta Astronautica*, vol. 60, no. 4-7 SPEC. ISS., pp. 341–350, 2007.

[15] S. Begum, M. Ahmed, P. Funk, N. Xiong, and B. V. Schéele, "Using Calibration and Fuzzification of Cases for Improved Diagnosis and Treatment of Stress," *Information and Computation*, pp. 93—-172, 1991.

[16] E. Jovanov, A. O. Lords, D. Raskovic, P. G. Cox, R. Adhami, and F. Andrasik, "Stress monitoring using a distributed wireless intelligent sensor system.," *IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine & Biology Society*, vol. 22, no. 3, pp. 49–55, 2003.

[17] A. de Santos Sierra, C. S. Ávila, J. G. Casanova, and G. B. del Pozo, "A stress-detection system based on physiological signals and fuzzy logic.," *IEEE Trans. Industrial Electronics*, vol. 58, no. 10, pp. 4857–4865, 2011.

[18] C. Lisetti and F. Nasoz, "Using noninvasive wearable computers to recognize human emotions from physiological signals," *EURASIP Journal on Applied Signal Processing*, pp. 1672–1687, 2004.

[19] I. Chatzigiannakis and S. Fischer, "WISEBED: an open large-scale wireless sensor network testbed," *Sensor Applications, ...*, pp. 68–87, 2010.

[20] C. Burin Des Rosiers, G. Chelius, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, and T. Noel, "SensLAB: Very large scale open wireless sensor network testbed," *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 90 LNICST, pp. 239–254, 2012.

[21] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks Vlado," *Proceedings of the second international workshop on Multi-hop ad hoc networks: from theory to reality - REALMAN '06*, no. May, p. 63, 2006.

[22] M. Pudane, E. Lavendelis, and M. A. Radin, "Human Emotional Behavior Simulation in Intelligent Agents: Processes and Architecture," *Procedia Computer Science*, vol. 104, no. December 2016, pp. 517–524, 2017.

[23] C. Becker-Asano, *WASABI: Affect simulation for agents with believable interactivity.* PhD thesis, Uni Bielefeld, 2008.

[24] Š. Korečko, T. Herich, and B. Sobota, "JBdiEmo - OCC model based emotional engine for Jadex BDI agent system," *SAMI 2014 - IEEE 12th International Symposium on Applied Machine Intelligence and Informatics, Proceedings*, no. 050, pp. 299–304, 2014.

[25] A. Ortony, G. Clore, and A. Collins, *Cognitive Structure of Emotions.* Cambridge University Press, 1988.

[26] M. Duggirala, M. Singh, H. Hayatnagarkar, S. Patel, and V. Balaraman, "Understanding impact of stress on workplace outcomes using an agent based simulation.," in *SummerSim* (F. D. Rango and J. L. Risco-Martín, eds.), p. 35, Society for Computer Simulation International / ACM DL, 2016.

[27] M. Page, M. P. Advisor, and D. Ashlock, *Agent-Based Modelling of Stress and Productivity Performance in the Workplace*. PhD thesis, University of Guelph, 2013.

[28] M. Cabanac, "What is emotion?," *Behavioural Processes*, vol. 60, pp. 69–83, 2002.

[29] K. R. Scherer, "What are emotion? And how can they be measured?," *Social Science Information Sur Les Sciences Sociales*, vol. 44, no. 4, pp. 695–729, 2005.

[30] J. F. Lehman, J. Laird, and P. Rosenbloom, "A gentle introduction to soar, an architecture for human cognition," in *In S. Sternberg and D. Scarborough (Eds), Invitation to Cognitive Science*, MIT Press, 1996.

[31] R. Plutchik, "Emotion: Theory, research and experience. volume 1. theories of emotion. edited by plutchikr. and kellermanh.. (pp. 399; illustrated; £19.00.) academic press: London. 1980.," *Psychological Medicine*, vol. 11, no. 1, p. 207–207, 1981.

[32] Wikipedia, "James-lange theory," 2017. [Online; accessed 21-May-2017].

[33] Wikipedia, "Cannon–bard theory," 2017. [Online; accessed 21-May-2017].

[34] S. Schachter and J. E. Singer, "Cognitive, social, and physiological determinants of emotional state," *Psychological Review*, vol. 69, no. 5, pp. 379–399, 1962.

[35] R. S. Lazarus, *Emotion and adaptation*. New York: Oxford, 1991.

[36] J. J. Prinz, *Gut Reactions: A Perceptual Theory of the Emotions*. Oxford University Press, 2004.

[37] L. J. D., *Feelings: The Perception of Self*. Oup Usa, 2007.

[38] M. Schröder, H. Pirker, and M. Lamolle, "First suggestions for an emotion annotation and representation language," in *Proceedings of LREC'06 workshop on corpora for research on emotion and affect*, pp. 88–92, 2006.

[39] "HUMAINE Emotion Annotation and Representation Language (EARL) x2014; emotion-research.net."

[40] J. Hastings, W. Ceusters, B. Smith, and K. Mulligan, "Dispositions and processes in the emotion ontology," in *Proceedings of the 2nd International Conference on Biomedical Ontology*, vol. 833 of *CEUR Workshop Proceedings*, pp. 71–78, International Conference on Biomedical Ontology, July 2011.

[41] J. F. Sánchez-Rada and C. A. Iglesias, "Onyx: A Linked Data Approach to Emotion Representation," *Information Processing & Management*, vol. 52, pp. 99–114, January 2016.

[42] R. Axtell and R. L. Axtell, "Why agents? on the varied motivations for agent computing in the social sciences," in *Working Paper 17, Center on Social and Economic Dynamics, Brookings Institution*, p. 17, 2000.

[43] A. E. Henninger, R. M. Jones, and E. Chown, "Behaviors that emerge from emotion and cognition: Implementation and evaluation of a symbolic-connectionist architecture," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, (New York, NY, USA), pp. 321–328, ACM, 2003.

[44] M. Bratman, *Intention, plans, and practical reason*. Cambridge, MA: Harvard University Press, 1987.

[45] D. Pereira, E. Oliveira, N. Moreira, and L. Sarmento, "Towards an architecture for emotional bdi agents," in *2005 portuguese conference on artificial intelligence*, pp. 40–46, Dec 2005.

[46] M. Ochs, N. Sabouret, and V. Corruble, "Simulation of the Dynamics of Non-Player Characters' Emotions and Social Relations in Games," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 1, pp. 281–297, Dec. 2009.

[47] M. S. El-Nasr, T. R. Ioerger, and J. Yen, "Peteei: A pet with evolving emotional intelligence," in *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, (New York, NY, USA), pp. 9–15, ACM, 1999.

[48] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *in International Conference on Complex Systems*, pp. 16–21, 2004.

[49] S. Luke, C. Cioffi-revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multi-agent simulation environment."

[50] F. Perez, B. E. Granger, and J. D. Hunter, "Python: An ecosystem for scientific computing," *Computing in Science and Engg.*, vol. 13, pp. 13–21, Mar. 2011.

[51] D. Masad and J. Kazil, "Mesa: An agent-based modeling framework," in *Proceedings of the 14th Python in Science Conference (SciPy 2015)*, pp. 53–61, 2015.

[52] E. Merino, "Soba: a simulator of occupancy based on agents," 2017.

[53] P. Aznar Delgado, "Design and Development of an Agent-based Social Simulation Visualization Tool for Indoor Crowd Analytics based on the library Three.js," Master's thesis, UPM, June 2017.

[54] H. c. Choi and S. y. Oh, "Realtime facial expression recognition using active appearance model and multilayer perceptron," in *2006 SICE-ICASE International Joint Conference*, pp. 5924–5927, Oct 2006.

[55] N. Sebe, M. S. Lew, Y. Sun, I. Cohen, T. Gevers, and T. S. Huang, "Authentic facial expression analysis," *Image Vision Comput.*, vol. 25, pp. 1856–1863, Dec. 2007.

[56] J. Sung and D. Kim, "Real-time facial expression recognition using staam and layered gda classifier," *Image Vision Comput.*, vol. 27, pp. 1313–1325, Aug. 2009.

[57] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study.," *Image Vision Comput.*, vol. 27, no. 6, pp. 803–816, 2009.

[58] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, (Washington, DC, USA), pp. 200–, IEEE Computer Society, 1998.

[59] S. Lalitha and S. Tripathi, "Emotion detection using perceptual based speech features," in *2016 IEEE Annual India Conference (INDICON)*, pp. 1–5, Dec 2016.

[60] T. L. Nwe, S. W. Foo, and L. C. D. Silva, "Speech emotion recognition using hidden markov models.," *Speech Communication*, vol. 41, no. 4, pp. 603–623, 2003.

[61] M. S. Unluturk, K. Oguz, and C. Atay, "Emotion recognition using neural networks," in *Proceedings of the 10th WSEAS International Conference on Neural Networks*, NN'09, (Stevens Point, Wisconsin, USA), pp. 82–85, World Scientific and Engineering Academy and Society (WSEAS), 2009.

[62] S. Kuchibhotla, B. S. Yalamanchili, H. D. Vankayalapati, and K. R. Anne, *Speech Emotion Recognition Using Regularized Discriminant Analysis*, pp. 363–369. Cham: Springer International Publishing, 2014.

[63] P. Shen, Z. Changjun, and X. Chen, "Automatic speech emotion recognition using support vector machine," in *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, vol. 2, pp. 621–625, Aug 2011.

[64] J. F. Sánchez-Rada, C. A. Iglesias, I. Corcuera-Platas, and O. Araque, "Senpy: A Pragmatic Linked Sentiment Analysis Framework," in *Proceedings DSAA 2016 Special Track on Emotion and Sentiment in Intelligent Systems and Big Social Data Analysis (SentISData)*, October 2016.

[65] M. M. Bradley and P. J. Lang, "Affective norms for English words (ANEW): Stimuli, instruction manual, and affective ratings," tech. rep., Center for Research in Psychophysiology, University of Florida, Gainesville, Florida, 1999.

[66] R. J. Davidson, "Affective neuroscience and psychophysiology: Toward a synthesis," *Psychophysiology*, vol. 40, no. 5, pp. 655–665, 2003.

[67] K. Ansari-Asl, G. Chanel, and T. Pun, *A Channel Selection Method for EEG Classification in Emotion Assessment Based on Synchronization Likelihood*, ch. 4. Zenodo, Sep 2007.

[68] P. J. Lang, M. K. Greenwald, M. M. Bradley, and A. Hamm, "Looking at pictures: Affective, facial, visceral, and behavioral reactions," *Psychophysiology*, vol. 30, no. 3, pp. 261–273, 1993.

[69] T. Rattanasawad, K. Saikaew, M. Buranarach, and T. Supnithi, "A review and comparison of rule languages and rule-based inference engines for the semantic web," in *Computer Science and Engineering Conference (ICSEC), 2013 International*, pp. 1–6, Sept 2013.

[70] B. De Meester, D. Arndt, P. Bonte, J. Bhatti, W. Dereuddre, R. Verborgh, F. Ongenae, F. De Turck, E. Mannens, and R. Van de Walle, "Event-driven rule-based reasoning using EYE," in *Joint Proceedings of the 1st Joint International Workshop on Semantic Sensor Networks and Terra Cognita and the 4th International Workshop on Ordering and Reasoning*, Oct. 2015.

[71] J.DeRoo, "Euler yet another proof engine." `http://eulersharp.sourceforge.net`. 2013.

[72] M. Coronado and C. A. Iglesias, "Task Automation Services: Automation for the masses," *Internet Computing, IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[73] O. Araque, "Design and implementation of an event rules web editor," trabajo fin de grado, Universidad Politécnica de Madrid, ETSI Telecomunicación, July 2014.

[74] S. J. E. Taylor, A. Khan, K. L. Morse, A. Tolk, L. Yilmaz, and J. Zander, "Grand challenges on the theory of modeling and simulation.," in *SpringSim (TMS-DEVS)* (G. A. Wainer, P. J. Mosterman, F. J. Barros, and G. Zacharewicz, eds.), p. 34, ACM, 2013.

[75] S. Lupien, B. McEwen, M. Gunnar, and C. Heim, "Effects of stress throughout the lifespan on the brain, behaviour and cognition," *Nature Reviews Neuroscience*, vol. 10, pp. 434–445, 6 2009.

[76] S. Marcora, W. Staiano, and V. Manning, "Mental fatigue impairs physical performance in humans," *Journal of applied psychology*, vol. 106, pp. 857–864, 2009.

[77] S. BG, "More realistic human behavior models for agents in virtual worlds: Emotion," 2001.

[78] C. P. Yaglou and D. Minaed, "Control of heat casualties at military training centers.," *Arch. Indust. Health*, vol. 16, no. 4, pp. 302–16, 1957.

[79] A. Rasdan, C. Mohammad, M. Hanifiah, and M. Haniff, "The Impact of Workers Productivity under Simulated Environmental Factor by Taguchi Analysis," *Procedia - Social and Behavioral Sciences*, vol. 10, pp. 263–268, 2014.

[80] E. Somanathan, R. Somanathan, A. Sudarshan, M. Tewari, E. Somanathan, R. Somanathan, A. Sudarshan, and M. Tewari, "The Impact of Temperature on Productivity and Labor Supply : Evidence from Indian Manufacturing The Impact of Temperature on Productivity and Labor Supply : Evidence from Indian Manufacturing ∗," *Discussion Papers in Economics*, no. August, 2015.

[81] M. Singh, M. Duggirala, H. Hayatnagarkar, and V. Balaraman, "A multi-agent model of work-group behaviour in an enterprise using a compositional approach," *CEUR Workshop Proceedings*, vol. 1561, no. ModSym, pp. 10–15, 2016.

[82] D. Abbott, "Calming the office cacophony: in the modern office environment the sources of noise, which can be defined very simply as unwanted sound, are many. the ways in which this noise and its effects on staff can be reduced are similarly numerous. duncan abbott goes through some of the main solutions.(noise and hearing)," *The Safety & Health Practitioner*, vol. 22, no. 1, p. 34(3), 2004-01-01.

[83] E. Gulian and J. R. Thomas, "The effects of noise, cognitive set and gender on mental arithmetic performance," *British Journal of Psychology*, vol. 77, no. 4, pp. 503–511, 1986.

[84] L. E. Maxwell, "Noise in the office workplace," *Department of Design and Environmental Analysis, College of Human Ecology, Cornell University*, 1999.

[85] M. V. Thoma, R. L. Marca, R. Brönnimann, L. Finkel, U. Ehlert, and M. Urs, "The Effect of Music on the Human Stress Response," *PLoS ONE*, vol. 8, no. 8, pp. 1–12, 2013.

[86] A. C. Jerejian, C. Reid, and C. S. Rees, "The contribution of email volume, email management strategies and propensity to worry in predicting email stress among academics," *Computers in Human Behavior*, vol. 29, no. 3, pp. 991 – 996, 2013.

[87] K. Kushlev and E. W. Dunn, "Checking email less frequently reduces stress," *Comput. Hum. Behav.*, vol. 43, pp. 220–228, Feb. 2015.

[88] K. Pattison, "Worker, interrupted: The cost of task switching," 2008.

[89] E. Merino, "Design and implementation of an Agent-based Social Simulation Model of Energy Related Occupant Behaviour in Buildings," tfg, ETSI Telecomunicación, Universidad Politécnica, Madrid, June 2017.

[90] N. J. N. P. E. Hart and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems, Science, and Cybernetics*, vol. SSC-4, no. 2, pp. 100–107, 1968.

[91] C. Sarkar, S. N. A. U. Nambi, R. V. Prasad, and A. Rahim, "A scalable distributed architecture towards unifying iot applications," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pp. 508–513, March 2014.

[92] S. Muñoz, "Development of a Task Automation Platform for Beacon enabled Smart Homes," Master's thesis, UPM, feb 2016.

[93] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[94] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[95] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[96] I. Goodfellow, D. Erhan, P.-L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio, "Challenges in representation learning: A report on three machine learning contests," 2013.

[97] M. Gallissot, J. Caelen, N. Bonnefond, B. Meillon, and S. Pons, "Using the Multicom Domus Dataset," Research Report RR-LIG-020, LIG, Grenoble, France, 2011.

[98] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio, "Challenges in Representation Learning: A report on three machine learning contests," *ArXiv e-prints*, July 2013.

[99] A. Gera and A. Bhattacharya, "Emotion recognition from audio and visual data using f-score based fusion," in *Proceedings of the 1st IKDD Conference on Data Sciences*, CoDS '14, (New York, NY, USA), pp. 2:1–2:10, ACM, 2014.

APPENDIX A

Emotion recogniser installation and usage

This Appendix gives a detailed explanation about the installation and usage of the emotion recogniser block. It is developed in Python, with the purpose of making it compatible with most computers. It is recommended to work in a virtual environment, that can be created and activated with Anaconda[1] using the following commands:

```
conda create -n emotion_detection python=3.4
source activate emotion_detection
```

The next step is the installation of dependencies. The following commands will install Scikit-learn, OpenCV, Keras, Pandas, H5PY, SpeechRecognition, PortAudio and PyAudio:

```
conda install scikit-learn
conda install -c menpo opencv3=3.1.0
pip install --upgrade keras
conda install pandas
conda install h5py
pip install SpeechRecognition
apt-get install portaudio19-dev
pip install pyaudio
```

[1] Anaconda download: https://www.continuum.io/downloads

Then, it is needed to configure Keras for working with Theano, create the file " /.keras/keras.json" with the following content:

```
{
    "image_dim_ordering": "th",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "theano"
}
```

Once the installation and configuration of the module has been completed, the usage is quite simple. The emotion detection from face analysis offers two possible ways of run the program: by mean of the command line or with a GUI.

For the command line option must be run the "face_analyser/run.py" script, that will start the emotion detection process will start, showing the output in the terminal.

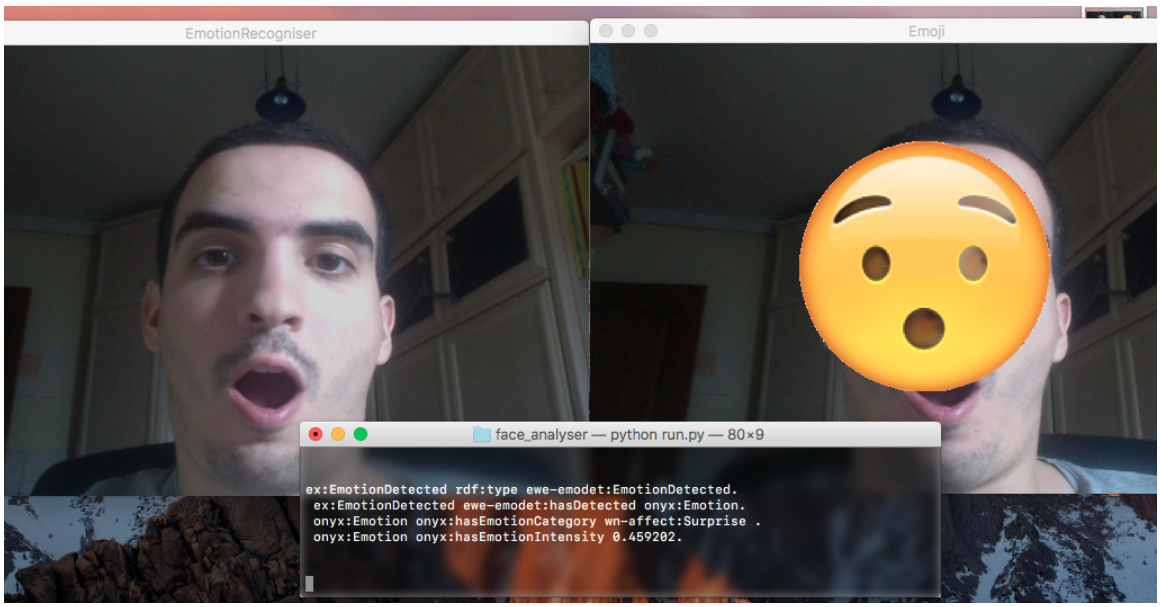

Figure A.1: Surprise emotion detection with the face analyser

The GUI option is launched by running "face_analyser/gui.py" script, showing a very simple user interface with five buttons:

1. Start emotion detection: starts the emotion recognition, printing the outputs in the terminal.

2. Train model: starts the model training with the data specified at "configuration/-data_settings.py".

3. Evaluate model: evaluates the model with the data specified at "configuration/-data_settings.py".

4. Get data from dataset: converts the data from the csv dataset to npy files.

5. Exit: closes the application.

With regards to the emotion detection from speech analysis, it can be launched by running "speech_analyser/run.py" script, that will start the emotion detection process, showing the output in the terminal.

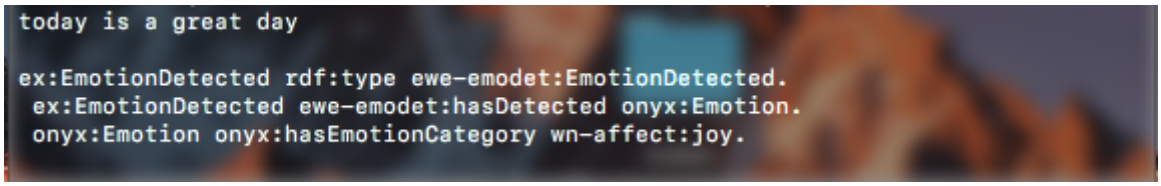

Figure A.2: Joy emotion detection with the speech analyser

APPENDIX B

Emotion simulator installation and usage

This Appendix details the step to follow about the installation of the emotion recogniser block. It is developed in Python, with the purpose of making it compatible with most computers. It is recommended to work in a virtual environment, that can be created and activated with Anaconda[1] using the following commands:

```
conda create -n smartoffice_simulation python=3.6
source activate smartoffice_simulation
```

The next step is the installation of dependencies. The following commands will install Mesa and Matplotlib. The former is the agent-based modeling framework, while the later is plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments, enabling the visualization of the simulation results:

```
pip install mesa
pip install matplotlib
```

Once the installation of the dependencies has been completed, the simulation must be configured, in order to test the required parameters. The different configuration files are described below.

[1] Anaconda download: https://www.continuum.io/downloads

Listing B.1 shows the general settings file, where the time in seconds that represents each step, the number of workers of the simulation and the number of steps to be run can be configured.

Listing B.1: General settings

```
time_by_step = 60 # seconds
number_of_workers = 10
number_of_steps = 6800
```

Aspects related to the model design, such as the contributions to stress of the different parameters evaluated are configured in "model_settings" file, shown in Listing B.2. The parameters that can be set are the contributions to stress of overtime hours, rest, email reception, ambient, noise, luminosity. In addition the task automation contribution attribute refers to the percentage of increase in the efficiency that provides the automation of tasks.

Listing B.2: Model design settings

```
overtime_contribution = 0.021
rest_time_contribution = 0.016
email_reception_contribution = 0.0029
ambient_contribution = 0.0012
noise_contribution = 0.03
luminosity_contribution = 0.000153
tasks_automation_contribution = 0.25
```

The automation policies are configured in "automation_settings" file, shown in Listing B.3.

Listing B.3: Workload settings

```
automate_temperature = False
automate_noise = False
automate_luminosity = False
automate_emails = False
automate_tasks = False
```

Workload settings file defines several aspects related to the workload, such as the estimated time to complete each task, the task arriving distribution, the emails read distribution and the time spent in reading each of them. As shown in Listing **??**, the emails read distribution params is chosen depending on if they are automated or not.

Listing B.4: Workload settings

```
import configuration.automation_settings as automation_settings

task_estimated_time = 25 # minutes
tasks_arriving_distribution_params = 20, 6

if automation_settings.automate_emails:
    emails_read_distribution_params = 4.70, 4.1
else:
    emails_read_distribution_params = 12.54, 8.02
email_read_time_distribution_params = 3, 0.5
```

Once configured the required values, the simulation can be started by mean of the run of “emotion_simulation/run.py” script.

APPENDIX C

Rules and channels creation

This appendix goes through the process of installation of EWE Tasker automation platform, as well as the creation of channels and rules for the case study explained in Section 5.2.

C.1 Installation

The integration of the platform with Docker[1] significantly facilitates the deployment of the task automation platform, that can be done with only four commands:

```
git clone https://github.com/gsi-upm/ewe-tasker.git
cd ewe-tasker
docker build -t gsiupm/ewetasker .
sh start.sh
```

Once run the above commands, the EWE Tasker platform has been deployed, and can be accesed from `http://localhost:8080`.

[1] Docker: https://www.docker.com

C.2 Channels definition

Once the platform has been deployed, the next step is the creation of the needed channels. For the case study presented in this master thesis, several channels are used. However, these channels are supposed to be created, as they represent services and devices integrated at previous work. For this reason, this section only focuses on the creation of the Emotion Detector channel and Google Home assistant.

- The channel Emotion Detector is created with the following params:
  - **Title:** emotiondetector.
  - **Description:** This channel represents an emotion detector.
  - **Nicename:** Emotion Detector.
  - **Emotion Detected Event:**

```
?event rdf:type ewe-emodet:EmotionDetectedWithIntensity.
?event ewe-emodet:hasDetected ?emotion.
?emotion onyx:hasEmotionCategory wn-affect:#PARAM_1# .
?emotion!onyx:hasEmotionIntensity math:greaterThan #PARAM_2#.
```

  - **Emotion Detected Event with certain intensity:**

```
?event rdf:type ewe-emodet:EmotionDetectedWithIntensity.
?event ewe-emodet:hasDetected ?emotion.
?emotion onyx:hasEmotionCategory wn-affect:#PARAM_1# .
?emotion!onyx:hasEmotionIntensity math:greaterThan #PARAM_2#.
```

  - **Emotion Detected Event by certain analyser:**

```
?event rdf:type ewe-emodet:EmotionDetectedWithIntensity.
?event ewe-emodet:detectedBy ewe-emodet:#PARAM_1#.
?event ewe-emodet:hasDetected ?emotion.
?emotion onyx:hasEmotionCategory wn-affect:#PARAM_2# .
```

- The channel Google Home assistant is created with the following params:
  - **Title:** googlehome.
  - **Description:** This channel represents the Google Home assistant.
  - **Nicename:** Google GHomo.
  - **Tell Joke Action:**

```
ewe-gh:GoogleHome rdf:type ewe-gh:TellJoke .
```

C.3 Rules definition

This section describes the process of rule definition. Rules page shows a list containing all rules that have been created. Each rule item shows relevant information such as the channels involved (represented by their image), the rule description, the creator, the place where the rule works, and the date when was created. Rules can be filtered by place. For creating a new rule, the user may press the "Create new rule" button, and will be redirected to the Rule Editor interface.

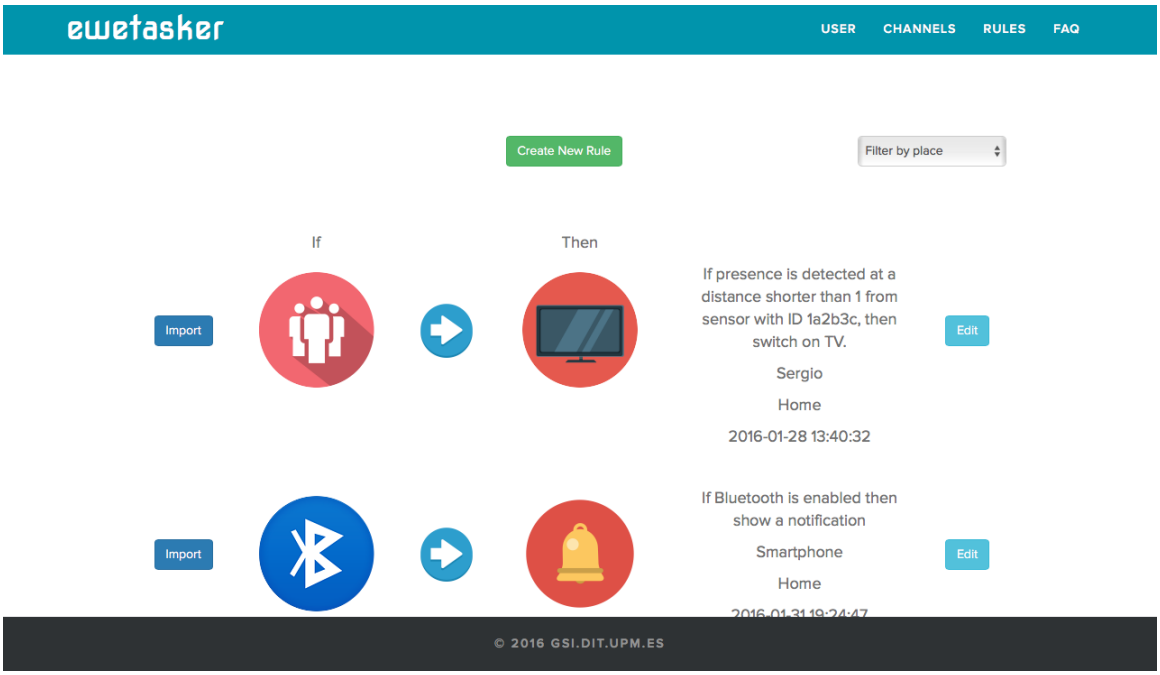

Figure C.1: Rules page

This interface has been designed to be easy to use and friendly to the user. The user selects the channel that will represent the event. This selection is made by dragging the icon representing the corresponding channel and dropping it into the left container. After this, a modal view appears containing all the events available for this channel. The user must select the wanted event and press the "OK" button.

Having configured the event of the rule, the user proceeds with the action. The action selection is very similar to the event selection, also made by dragging the icon representing the corresponding channel and dropping it into the right container. In the same way that the event, a modal view appears. In this case, it offers the actions available for the selected channel. The user selects the action which he wants to be triggered and press "OK". Once these configurations have been made, the rule can be saved by pressing the "Save Rule"

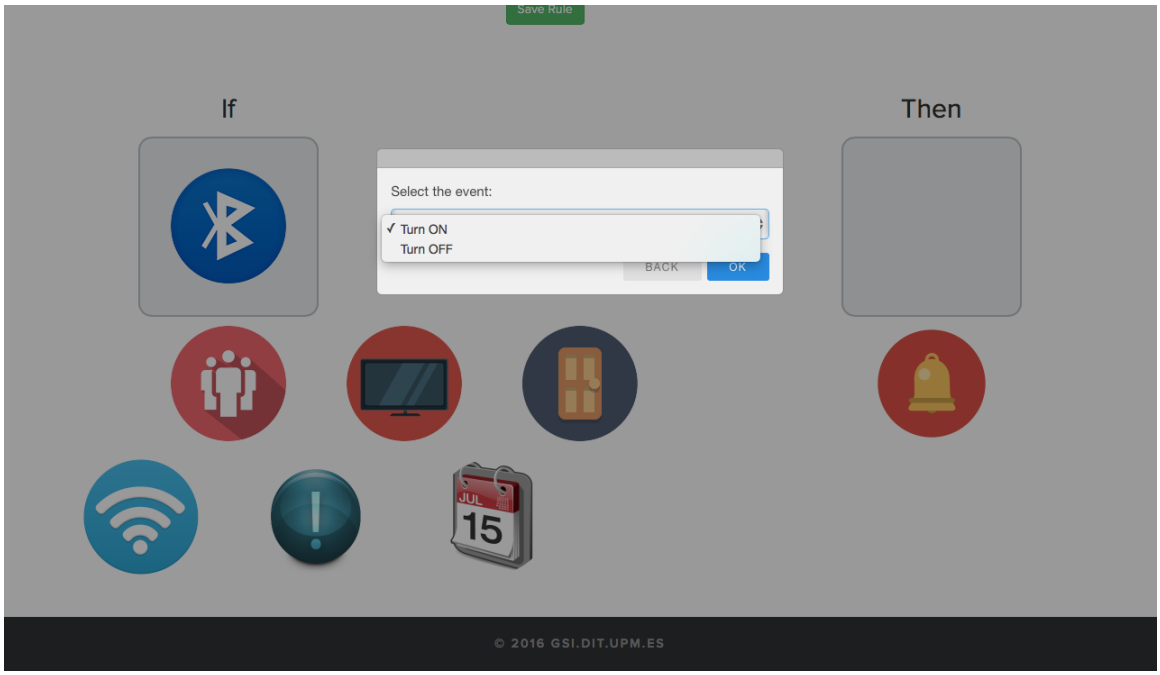

Figure C.2: Event Selection

button.

Now, a new modal view appears. This modal view contains several fields that must be filled. These fields represents rules attributes such as title, description, creator, place and all the params to configure for the events and actions (if any). Once these fields have been filled, the user may press "OK" and the rule will be stored.

At this moment, the automation configuration process has been completed, and the created rules will be evaluated along with the generated events for triggering the corresponding actions.

APPENDIX D

Model implementation with Keras

This Appendix gives a detailed explanation about the implementation of an adaptation of VGG16 model in Python using Keras, describing each used function. The first step is the creation of the model, using the Sequential constructor of Keras, that represents a model with a linear stack of layers.

The model needs to know what input shape it should expect. For this reason, the first layer in a Sequential model (and only the first, because following layers can do automatic shape inference) needs to receive information about its input shape. This is done by passing an *input_shape* argument to the first layer. This is a shape tuple (a tuple of integers or None entries, where None indicates that any positive integer may be expected). In order to provide the spatial padding of convolutional layer input, that provides control of the output volume spatial size and guarantees that the spatial resolution is preserved after convolution, the ZeroPadding2D layer is added. This layer can add rows and columns of zeros at the top, bottom, left and right side of an image tensor, and its constructor receives as parameter the padding.

The convolutional layer are implemented by mean of Conv2D constructor, that creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The most relevant arguments are the number of filters, that represents the dimensionality of

the output space; the kernel_size, that specifies the width and height of the 2D convolution window; the strides of the convolution along the width and height; and the activation function to use (relu in this case).

Listing D.1: Model implementation

```
self.model = Sequential()
self.model.add(ZeroPadding2D((1,1), input_shape=(1, 48, 48)))
self.model.add(Conv2D(32, (3, 3), activation="relu"))
self.model.add(ZeroPadding2D((1,1)))
self.model.add(Conv2D(32, (3, 3), activation="relu"))
self.model.add(MaxPooling2D((2,2), strides=(2,2)))

self.model.add(ZeroPadding2D((1,1)))
self.model.add(Conv2D(64, (3, 3), activation="relu"))
self.model.add(ZeroPadding2D((1,1)))
self.model.add(Conv2D(64, (3, 3), activation="relu"))
self.model.add(MaxPooling2D((2,2), strides=(2,2)))

self.model.add(ZeroPadding2D((1,1)))
self.model.add(Conv2D(128, (3, 3), activation="relu"))
self.model.add(ZeroPadding2D((1,1)))
self.model.add(Conv2D(128, (3, 3), activation="relu"))
self.model.add(ZeroPadding2D((1,1)))
self.model.add(Conv2D(128, (3, 3), activation="relu"))
self.model.add(MaxPooling2D((2,2), strides=(2,2)))

self.model.add(Flatten())
self.model.add(Dense(1024, activation='relu'))
self.model.add(Dropout(0.5))
self.model.add(Dense(512, activation='relu'))
self.model.add(Dropout(0.5))
self.model.add(Dense(6, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', \
    metrics=['accuracy'])
```

In addition, pooling layers are added by mean of MaxPooling2D constructor, which specifying pool_size (factors by which to downscale), strides and padding provides a max pooling operation for spatial data. Finally, FC layers are added with Dense constructor, that adds a regular densely-connected NN layer specifying the dimensionality of the output space and the activation function. Note that before adding the FC layers, have been added a flatten layer, that flattens the input without affecting the batch size and two dropout

layers, that consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

Finally, the model is compiled using the compile function, in order to configure the learning process by mean of three arguments: the optimizer, the loss function (the objective that the model will try to minimize) and a list of metrics.