FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

An Extensible Framework for Smart Environment Simulations for IoT

Renato Sampaio de Abreu

DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Hugo Sereno Ferreira

Co-Supervisor: André Restivo

July 21, 2019

An Extensible Framework for Smart Environment Simulations for IoT

Renato Sampaio de Abreu

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Filipe Alexandre Pais de Figueiredo Correia

External Examiner: Ângelo Manuel Rego e Silva Martins

Supervisor: Hugo José Sereno Lopes Ferreira

July 21, 2019

Abstract

The Internet of Things (IoT), refers to a paradigm, based on the concept of a pervasive presence around us of a variety of things or objects. The IoT can be applied in several aspects of our everyday life, enabling the creation of an intricate system of connected devices, which make our environment smarter, more measurable, manageable, and user-friendly. Nowadays, a plethora of commercially available IoT devices are pervasively used for home automation by non-tech savvy users, however, to achieve a connected and interactive environment, the design of a smart home is (a) hard to set up, (b) error-prone and (c) expensive.

Simulation tools provide the capability to overcome challenges faced by projects that envision the creation of a smart home environment. The inherent flexibility provides users the possibility to iteratively validate and test an endless set of modeling scenarios similar to real life environments, abstracting away physical challenges. While on the other hand, simulators have the potential to provide researchers with rich datasets for the testing of novel data analysis approaches, particularly at the early stages of development.

Simulators have evolved mainly in two directions: model-based and interactive approaches. Model-based approaches resort to the modeling of the simulation context, such as agents, activities, and processes, and as a result, the correctness of the simulation depends on the accuracy of the underlying activity model. Interactive based approaches provide an interface for the users to control and capture the context in real time, which consequently increases the granularity detail of the simulation context but in general, the overall process is a cumbersome task for the users.

Researchers and homeowners alike need a platform to experiment, test, and validate smart environments, whether with the purpose to generate reliable datasets or analyze the design of their smart houses. Although in recent years several attempts have been made, none of them is considered a comprehensive solution or was widely adopted. In general, the tools fail to leverage the advantages of both approaches and, at the same time, provide a flexible and extensible platform for the configuration of smart environments, with seamless integration between the surrounding context, the devices behavior, and agents interaction.

This project aims to pave the way for the development of realistic simulators capable of providing an experimental environment for users to test and analyze their smart homes, abstracting the technical details of implementing and managing an IoT system. The simulator provides the configuration of the domain problem as the users deem fit, without imposing a high number of constraints or wide the gap between the simulated scenario context and the real one.

The simulator integrates a (1) flexible device configuration, (2) extensive scenario, and context customization, (3) interactiveness and data generation, (4) simulation of physical properties and (5) straightforward agent modeling — the set of features collectively distinguish the simulator as a tool with a broad application range with the capability to design and experiment a wide variety of virtual scenarios.

Resumo

A Internet das Coisas (IoT) refere-se a um paradigma, baseado na presença pervasiva de dispositivos. A IoT é aplicada a vários aspetos do nosso quotidiano, permitindo a criação de um sistema de dispositivos interligados, que tornam o ambiente adjacente mais inteligente e administrável. Hoje em dia, uma grande quantidade de dispositivos IoT são utilizados na criação de casas inteligentes e inerente automação. Contudo, o design de uma casa inteligente é (a) difícil de configurar, (b) propenso a erros e (c) caro.

As ferramentas de simulação tem a capacidade de ultrapassar os desafios encontrados em projetos que visam o design e criação de um casa inteligente. A flexibilidade destas ferramentas fornece aos utilizadores a possibilidade de validar iterativamente e testar um conjunto interminável de cenários de modelação semelhantes a contextos da vida real.

Ao longo dos anos, os simuladores aplicados a casas inteligentes evoluíram principalmente em duas direções: abordagens baseadas em modelação e abordagens interativas. Abordagens baseadas em modelos recorrem à modelação do contexto, tal como os agentes, atividades e processos, e por conseguinte, a qualidade da simulação está diretamente relacionada com o modelo desenvolvido. Abordagens interativas incorporam uma interface para os utilizadores controlarem e capturarem o contexto em tempo real, o que aumenta a granularidade e qualidade do contexto de simulação, contudo, o processo é moroso.

A criação de um ambiente inteligente é facilitada com o uso de uma plataforma para experimentar, testar e validar os respetivos contextos e o design da casa. Embora nos últimos anos várias tentativas tenham sido feitas, nenhuma ferramenta é considerada uma solução abrangente. Em geral, as ferramentas não incorporam as vantagens de ambas as abordagens, fornecendo ao mesmo tempo uma plataforma flexível e extensível para a configuração de casas virtuais inteligentes, com integração consistente entre o contexto envolvemente, o comportamento dos dispositivos e a interação dos agentes.

Este projeto apresenta-se como uma ferramenta de simulação com a capacidade de representar o contexto de casas virtuais, abstraindo os detalhes técnicos de implementação e manutenção de um sistema IoT. O simulador permite a configuração do domínio e contexto conforme os requerimentos do utilizador, sem impor demasiadas restrições de modelação ou aumentar as divergências entre o contexto do cenário simulado e o real.

O simulador integra as seguintes funcionalidades: (1) configuração de dispositivos flexível, (2) criação de cenários extensível, (3) interatividade e geração de dados, (4) simulação de propriedades físicas e (5) modelação do comportamento de agentes - este conjunto de funcionalidades distingue o simulador como uma ferramenta com uma vasta aplicação que permite criar e experimentar uma grande variedade de cenários virtuais.

Acknowledgements

I would first like to thank my thesis advisors Hugo Sereno, and André Restivo of the Faculdade de Engenharia da Universidade do Porto. Whenever a question arose, or I found myself in a trouble spot regarding the research and project development, they consistently steered me in the right direction.

Finally, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Renato Abreu

"A ship in harbor is safe,
but that is not what ships are built for."

John A. Shedd

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ADL	Activiy of Daily Living
AI	Artificial Intelligence
AmI	Ambient Intelligence
CA	Cellular Automaton
GUI	Graphical User Interface
IE	Intelligent Environment
IoT	Internet of Things
SH	Smart Home
UI	User Interface
WBS	Web-based Simulation
WWW	World Wide Web

Chapter 1

Introduction

Section 1.1 introduces a general overview surrounding the management and simulation of Internet of Things environments. Section 1.2 (p. 2) explains the reasoning of why it would be useful to have a solution to the problem defined in this dissertation. Section 1.3 (p. 2) defines the problem that leads to the work developed in this area. Section 1.4 (p. 3) specifies the expected outcomes and goals that the project ought to achieve. Finally, Section 1.5 (p. 4) describes the structure and chapters that are part of this report.

1.1 Context

Feng Xia et al. [XYWV12] defined the Internet of Things, IoT, as the networked interconnection of everyday objects equipped with ubiquitous intelligence. The pervasiveness of these smart objects around us creates an interactive environment between things and humans.

The key characteristics of the Internet of Things — communication, identification, and interactivity — alongside with the main features of an IoT system foster a plethora of opportunities in the fields of Home Automation, Transportation, Manufacturing, Healthcare, Farming, and Retail [MSPC12].

The massive proliferation of the Internet of Things is undergoing, with the number of connected devices expected to reach 20.4 billion by 2020, as predicted by Gartner [vdM17]. The assumptions presented by multiple business companies point the IoT towards the same direction: an ever-growing presence of pervasive devices in the most varied environments. This presence correlates directly to a more significant influence on people's lives and massive amounts of data produced.

Intelligent Environments (IEs) are environments capable of monitoring their context and the conditions of their inhabitants to enhance the overall experience within that environment [NPF+11]. The environment integrates several sensor technologies capable of data collection, with configurations dependent on the specific monitoring domain and the needs of its inhabitants.

As the “things” play a more prominent role in our daily lives, the overwhelming number of connected devices around us increases the difficulty to manage them. For the end-users, there are two main approaches to solving the management of IoT devices: Virtual Programming Platforms and Intelligent Assistants.

On the other hand, for experienced users, researchers, and architects, the design of a smart environment is no trivial task, so the design must be thoroughly planned with the goal of diminishing future problems. Simulators appear as a perfect solution to such scenarios because of the possibility to recreate complex scenarios, system operations, and models.

Currently, the analysis of data captured from smart environments is increasingly being used to provide better services for users. The Artificial Intelligence research field appears as an enhancer for the IoT, enabling the development of machine learning techniques that use the data gathered by physical and simulated smart devices.

1.2 Motivation

Researchers and homeowners alike need a platform to experiment, test, and validate smart environments, whether with the purpose to generate reliable datasets or analyze the design of their smart houses.

For the target audience specified above, the creation of a smart home is an expensive investment and challenging task. When we take into account setbacks such as device inadequacy for the domain, improper device use, or configuration, the risks of designing a smart house have to be thoroughly analyzed to diminish costs and future problems [SCNM14].

Smart home environments intend to ease the living conditions of its inhabitants; however, it is difficult for an end-user to analyze and validate how the overall system reacts to all the possible contexts. After all, a smart home is a "live" environment where the occurrence of erratic behaviors is nothing out of extraordinary due to the presence of humans.

The validation and testing of IoT systems when in a heterogeneous and complex ecosystem like a smart home has several challenges. Thus far, the solutions available for testing these systems are insufficient and fragmentary [DCPF18]. Nevertheless, the analysis of the virtual smart home under erratic or unexpected behavior can prevent the eventual impact on people’s lives.

1.3 Problem Definition

Several research areas, such as the classification and recognition of activities of daily living (ADL), anomaly detection in elderly daily behavior and optimization of electrical consumption, per example, when applied to machine learning techniques entail the need for datasets that allow testing and validation of the results. These research areas require datasets, synthetic, or real, with high reliability gathered from smart environments scenarios. The generation of datasets has two main categories: datasets generated from real smart homes testbeds or smart home simulation tools.

Real testbeds have more reliable results compared to simulation tools; nevertheless, costs of setup and maintenance appear as an essential barrier for their wide adoption.

On the other hand, users, while building a smart environment, come upon many challenges: optimal placement of sensors, lack of devices standardization, privacy issues, and integrating middleware to act as a bridge between devices. The solution to these problems is crucial to the building of an efficient and optimized system; however, currently, there is a clear gap regarding the tools that can effectively provide an environment for the users to experiment and tamper with the design of a smart home.

The development of a pervasive computing system requires a fully assembled computing environment to test and run the application accurately, which represents the system's first test iteration — the simulation models the physical environment and system processes. The simulation is a step necessary to ensure that most of the problems that arise are minimal and the smart home design is optimal. The analysis of the results from this first iteration can reveal the need to redesign a smart home by adding or removing smart devices, change the placement, or change the scenario settings. In a real smart home, if such a situation happens, the solution is usually costly or infeasible to carry out, whereas, with a virtual smart home, the simulation provides tools to solve the problems quickly.

A smart home is a "live" environment, where the possibilities are endless. The ever-growing number of devices available in the market increases the scenarios that a user can design within their home. The heterogeneity of devices and lack of standardization in their representation increases the complexity to simulate the plethora of devices and their behaviors appropriately.

Also, the thriving and lively style of a human being together with the dynamism of the physical environment poses a challenge for the modeling and simulation of scenarios present in a smart home. As pointed out, smart environment simulators usually have to address and successfully overcome the challenges extremely complex in nature, which has severely hindered the development of tools alike.

Although in recent years several attempts have been made in terms of the development of simulation tools, none of them is considered a comprehensive solution or was widely adopted. The primary reason lies in the fact that the approaches are domain restricted to the problem at hand and often integrate a limited and inflexible representation of a real smart environment.

1.4 Goals

The goal of this project is twofold: First, ease the validation and design of smart environments by providing a simulator capable of recreating a multitude of scenarios and fundamental interactions. Second, the acquisition of reliable datasets to apply in machine learning techniques or posterior analysis of the smart environment context.

A simulation tool will be developed with the capability to recreate complex smart environment scenarios with fine-grained details. The primary purpose of this simulator, having flexibility as the

driven force, is to ease the replication of the most diverse set of scenarios with high fidelity without relinquishing usability.

As flexibility is one of the key features of this project, users should be able to create virtual environments with different degrees of complexity:

- **Agents**: (1) The needs of an inhabitant and the inherent complex behaviors and (2) the activities, their time-span, the effects on the environment and on the agents;
- **Devices**: (3) The configuration of devices, properties, and actions and (4) the interaction and behaviors established between devices;
- **Layout**: (5) Fully customizable layout with a graphical visualization and (6) specification of environment settings and modeling of physical properties.

In this dissertation, we analyze the development of an extensive and flexible architecture, incorporating the pivotal features of a smart home environment. This approach is a starting point for the development of realistic simulators with a flexible configuration of scenarios. Besides, the project aims to incorporate an architecture with the capability to support new requirements without becoming obsolete. Although this principle is subjective to the requirements in mind, the goal is to provide future-proofing for the primary features of this simulator.

1.5 Dissertation Structure

Besides the introduction, this dissertation contains the following chapters: Chapter 2 (p. 5) describes the state-of-the-art, mainly divided in three parts: An in-depth background to the Internet of Things, work related to the management of IoT systems and tools developed for the simulation of smart homes. Chapter 3 (p. 23) details the current shortcomings of the state-of-the-art and proposes a solution to solve, or at least diminish, the problem in question. Chapter 4 (p. 29) explains the architecture and main components of the simulator developed and specific architectural and technological choices made throughout the development. Chapter 5 (p. 49) describes the evaluation procedure used to ensure that the simulator developed meets the goals initially defined. Chapter 6 (p. 59) summarizes the work developed, the contributions to the field, and future improvements.

Chapter 2

Literature Review

This chapter thoroughly describes the review of the state-of-the-art for management and simulation tools for the Internet of Things environments.

Section 2.1 and respective subsections provide insights and background information about the Internet of Things and its central concepts, essential to understanding correctly the problem analyzed in this dissertation. Section 2.2 (p. 8) overviews the current tools used for the management of IoT systems, detailing its shortcomings and strengths. Section 2.3 (p. 12) focuses on tools for modeling and simulation of IoT environments, features, and principal purposes. Lastly, Section 2.4 (p. 19) outlines and summarizes the current state-of-the-art.

2.1 Internet of Things

The Internet of Things (IoT), refers to a novel paradigm, with the concept of pervasive presence around us of a variety of things or objects, which, through unique addressing schemes, can interact with each other and cooperate with their neighbors to reach common goals [AIM10].

The IoT can be equally applied in several aspects of our everyday life or large-scale scenarios, enabling the creation of an intricate system of connected devices, which, on its own, make our environment smarter, more measurable, manageable and user-friendly. The adoption and presence of smart devices in our daily life are increasing at an incredible rate, having applications in the most diverse fields, such as Home Automation, Transportation, Manufacturing, Healthcare, Farming, Retail, among others [MSPC12].

Gartner estimates that by 2020 there will be approximately 20.4 connected "things" [vdM17]. Among these, more than half of the applications are expected to target the consumer segment. Hence, the role of IoT in our daily life is particularly disruptive.

2.1.1 Home Automation

Home automation technology refers to the interconnection of appliances and devices such as lights, temperature and humidity sensors, TVs, power plugs, security, and fire systems. Devices like smartphones, desktops, or laptops usually allow the remote control of the home automation systems [ROL+13]. The main goal of such systems is to provide comfort, convenience, and increase the quality of life and security for the end-users [AR16].

The term smart home is an application of pervasive computing. Nowadays, the use of terms such as a smart house, domotique, intelligent home, and home automation is interchangeable. While home automation refers to the devices within the home that eases its control and monitoring, a smart home is an application of pervasive computing "that can provide user context-aware automated in the form of ambient intelligence, remote home control or home automation," as defined by current trends in smart home research [ARA12].

The interpretation and analysis of the environment incorporate the smartness and context awareness in the services provided by the smart homes. If the appliances and devices can respond and adapt their settings to human behavior automatically, in a non-obtrusive way, and always according to the inhabitant requirement, it provides ambiance intelligence, which brings more automation to the user's home [ERA09].

Nowadays, the setup of these devices in a smart home environment and access to its capabilities, even with the use of conversational systems or graphical interfaces, require some knowledge and hands-on work, which inherently, is a problem for non-tech savvy users [SDPA14].

2.1.2 Cloud & Fog Computing

As defined by National Institute of Standards and Technology (NIST), "Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources" [MG11] provided by a service provider. The cloud computing model has five essential characteristics:

- **On-demand self-service** - Services provided as needed, seamlessly, without the need for manual requests to the service providers;
- **Broad network access** - Resources provided through common and interoperable protocols over the network;
- **Measured service** - The service provider controls and optimizes the resources usage. This usage is controlled, measured, and provided to the consumer;
- **Rapid elasticity** - The resources are automatically scaled and released according to demand; meanwhile, this process is abstracted to the consumer;
- **Resource pooling** - The resources are pooled by several consumers, whereas they automatically share the resources amongst themselves.

Literature refers to Cloud Computing as the facilitator of data processing and storage for IoT, because of all the possibilities that offer to these systems, such as data mining, real-time processing, and batch processing. The analysis of Table 2.1 summarizes differences between both concepts and enables us to understand how IoT complements Cloud Computing and vice versa. The IoT devices create vast quantities of data and have a pervasive presence, with limited reachability and storage, which creates a challenge to integrate applications that require high computational power. The Cloud provides a centralized service with easy access and virtually unlimited storage that can store and process the data gathered by the IoT devices.

Table 2.1: IoT and Cloud Computing comparison

	IoT	**Cloud**
Displacement	*Pervasive*	*Centralized*
Reachability	*Limited*	*Ubiquitous*
Role of the Internet	*Point of convergence*	*Means for delivering service*
Storage	*Limited or none*	*Virtually unlimited*

The Cloud acts as a point of monitoring and control of the smart home, as the devices and services should be accessed and manageable anytime and anywhere. The cloud-oriented services that interconnect the appliances within the home ease the automation aspect of a smart home [BDPP15].

As defined by the OpenFog Consortium, "Fog Computing is a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things.". The concept emerges to overcome the drawbacks encountered in the cloud computing architecture, mainly the high latency, low data transfer speed, and neglecting the computational and storage resources available in the IoT devices. Thus, Fog Computing offers a decentralized architecture where it is possible to execute IoT services closer to the devices (i.e., at the edge of the network), as seen in the Figure 2.1 (p. 8).

With a fog architecture, fog nodes at the edge ingest the data generated by appliances and other devices. This first level of the Fog, designed for machine-to-machine (M2M) interaction, collects, process the data, and issues control commands to the actuators. The data at this level can be processed, with real-time analytical processes, or sent to the higher levels, i.e., the Cloud. The second and third tiers deal with visualization and reporting (human-to-machine interactions), and systems and processes [BMZA12].

IoT brings intelligence into home devices and facilitates the measurement, control, and monitoring of the smart home environment. Cloud computing provides scalable computing and storage capacity for tasks that require intensive processing power that is usually not present in IoT devices [SAH+13]. Fog Computing allocates processing power closer to the edge of the network and provides a bridge of communication between the devices and the Cloud. The architectural layout of these concepts facilitates the integration of data analytics services that can apply machine learning techniques to the real-time and historical data gathered by the devices and gain insights from it.

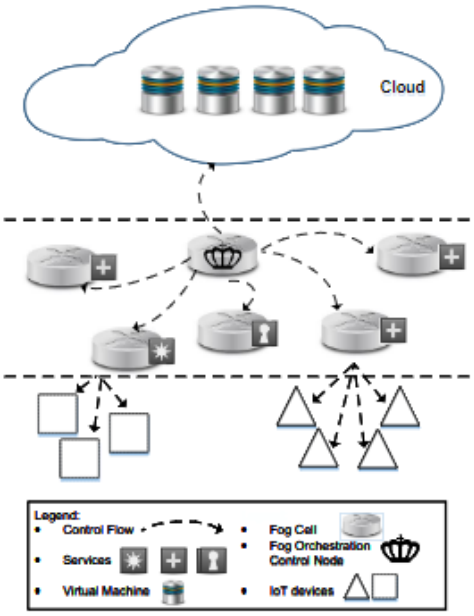


Figure 2.1: Fog architecture

2.2 Management of IoT Systems

With the ever-growing adoption of IoT devices, it is only natural that their management within our environment becomes complex. As the number of devices integrated increases, and their interactions become intricate, the difficulty to control the network of devices raises substantially.

The development of IoT management systems occurred due to the primary challenges that IoT systems face: (1) Technology-Centric Challenges and (2) Human-Centric Challenges [KH16]. The technology-centric challenges of IoT are:

- **Data Management:** the storage, processing, and analysis of the data acquired with the "things." The sheer amount of data generated by the devices, with non-uniform formats, is a major challenge for the IoT;
- **Device and Application Management:** The IoT devices do not have uniform standards, which makes them difficult to interoperate within the system;
- **Bridging Data across Platforms and Services:** The technological stack of an IoT platform encompasses several platforms for different purposes. The challenge lies in bridging the platforms and share data throughout the stack;
- **Search and Discoverability:** The "things" that are part of the network must be discoverable and identifiable;
- **Monitoring and Reporting:** The devices, independently of the context that they convey, must be monitored in real-time.

The human-centric challenges of IoT are:

- **Cognitive Burden :** The end-users must be educated regarding the capabilities and functionalities of devices. The way to obtain knowledge about the system, independently of the complexity, must be transparent;
- **User Interaction :** There must be an interface where the user can interact with the system and respective devices, in an interactive and user-friendly way;
- **Configuration Challenges :** The heterogeneity of devices ends up being a burden for the end-users in terms of the configuration since each device presents a unique way to be configured and used.

Gradually, the development of software projects intended to manage these IoT systems started to appear in the form of Visual Programming Platforms (VPP), which provide a user-friendly management overview of the system and simplify the process of managing a smart system. However, these platforms target mostly users with a background in the IoT context, having higher technical skills.

Work developed by Dias et al., show how the integration of concepts such as visual programming and liveness reduces the complexity of creating, managing, and monitoring of IoT systems, by increasing the feedback loop between the developer and the target system [DFF18].

Nevertheless, VPPs require the learning of how the software works and the respective logic, which no matter how simple or user-friendly it can appear, continues to entail specific barriers for the users, and as a result, few learn and accustom themselves to this kind of software. Chatbots and Intelligent Assistants appear as the solution for the problems mentioned above, offering a simple way to manage the IoT devices, through interactive and user-friendly voice and text interactions.

The following sections broadly explain the two types of approaches for the management of IoT systems: Visual Programming Platforms and Chatbots and Intelligent Assistants.

2.2.1 Visual Programming Platforms

Visual programming tools provide a user-friendly management overview. Whether they are created specifically for programming sensors or because they are general-purpose platforms compatible with those devices, these platforms abstract away the code and inner low-level configurations, which are the difficult aspects for non-technical users. Instead, they provide a simple graphical user interface based, normally, on drag-and-drop, toggles and widgets, that represent in more inclusive detail the underlying system, its settings, interactions, and allow the users to overview the system, manage it and configure the device's behavior and interactions. Regarding VPPs applied to the home automation context, Home Assistant[1], domoticz[2], and Node-RED[3] are the open-source products with larger adoption.

[1] https://www.home-assistant.io/
[2] https://github.com/domoticz/domoticz
[3] https://nodered.org/

Home Assistant is a home automation platform capable of tracking and controlling all devices in the IoT system. This platform allows the integration of devices and the definition of intricate automation settings with YAML configuration file that specifies the essential information and device settings.

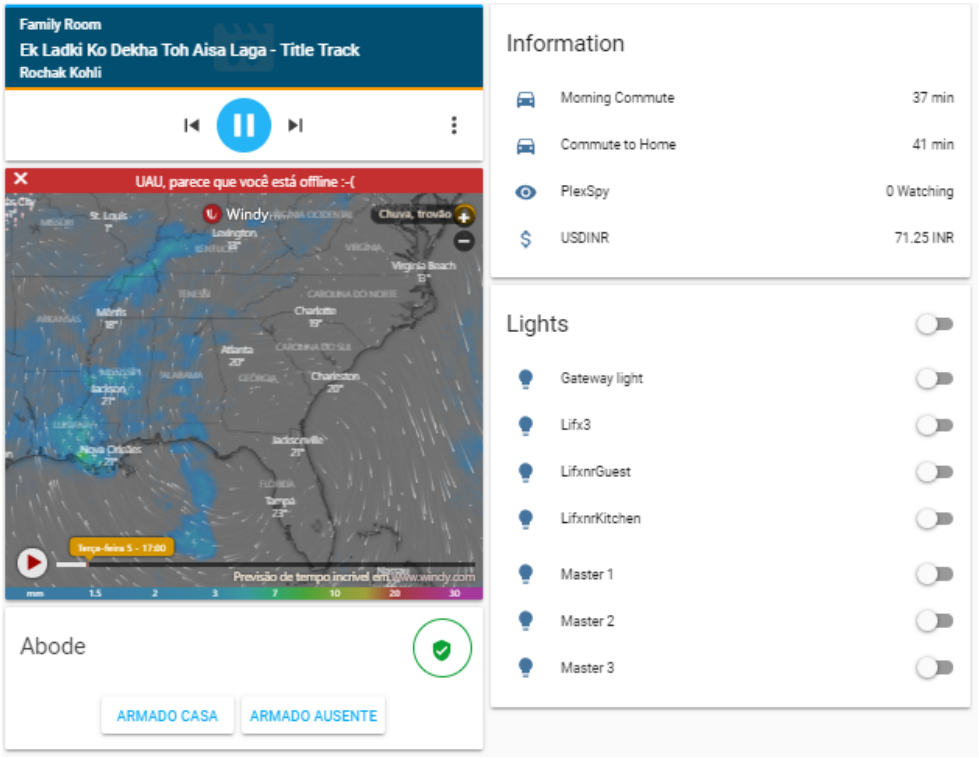


Figure 2.2: Home Assistant UI

Home Assistant offers a plethora of integrations that facilitate its setup, configuration, usage, and configuration plugins for the most varied use-cases, provided by the community. The integrated iOS application and web-based UI, as seen by the Figure 2.2, allow the users to configure the devices' state and analyze the data gathered. The management provided by Home Assistant focuses on simple toggles or selection inputs that define the current state of the devices, whereas a user-friendly UI supports the monitoring with the system overview.

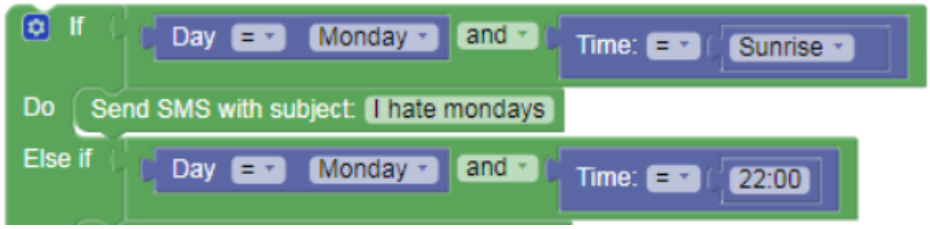


Figure 2.3: Flow-based GUI

Domoticz is a home automation system capable of monitoring and configuring IoT devices.

It offers a UI adapted to desktop and mobile devices, with an overview of the connected devices, where it is possible to configure them, analyze the data received through graphical visualizations and define custom push notifications to alert the user about specific events. Domoticz, in terms of exact automation features, allows the use of scripts or a flow-based GUI, where it is possible to define the content of the scripts, the respective trigger and their action on the system. The Figure 2.3 (p. 10) shows Domoticz flow-based GUI.

Node-RED is a flow-based programming tool for the Internet of Things, part of the JS Foundation. The flow-based programming implemented by Node-RED describes the system's behavior as a network of nodes, abstracting the low-level configurations and the code that establish the interaction between network entities. Visual representations are well suited for this model of programming, where the analysis of the system's flow is facilitated and understandable, as Figure 2.4 shows.

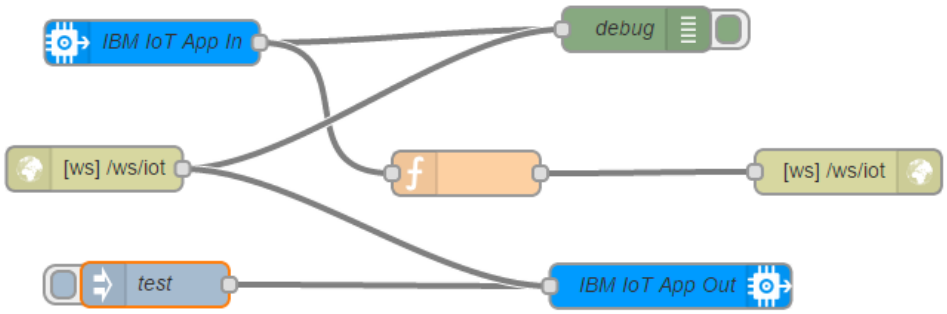


Figure 2.4: Node-RED flow-based programming

Node-RED has as basis Node.js, taking full advantage of its event-driven, non-blocking model. The lightweight core of Node-RED makes the tool ideal to run at the edge of the network on low-cost hardware and in the Cloud, whereas, through a browser, it is possible to access the tool editor. The browser-based flow editor, which offers a wide range of nodes in the palette, allows quick prototyping of the flows established between the hardware devices, APIs, and online services. The user can subsequently deploy the flows to the runtime environment.

Nodes are the basic building blocks for creating flows, representing logic abstractions of code. A collection of nodes establishes a flow that transmits messages between them. With Node-RED and its integral elements, it is possible to create complex flows that define intricate interactions and dependencies between devices, define complex rules and configurations, and, as a result, create a complex environment of autonomous IoT devices.

The flows created in Node-RED are stored using JSON, and the built-in library stores functions, templates, or flows. This extensibility makes it perfect for the community to share and contribute to the ever-growing repository of flows provided by Node-RED. The tool also offers integrations with well known message-middleware software, such as RabbitMQ, and data visualization platforms like Kibana, which allow to get more insights about the data being gathered and transmitted by the IoT system.

2.2.2 Chatbots

According to the literature, Chatbots have several terms and definitions associated with them. Intelligent Personal Assistants, Virtual Agents, Software agents, and Chatbots are interchangeable terms encompassed under the umbrella term "Agents," which refers to "A self-contained, interactive and concurrently executing object, possessing internal state and communication capability" [Hew77].

Chatbots with Artificial Intelligence imbued can understand natural language, maintain a context of the conversation flow, and overall appear intelligent. Those are the denominated Intelligent Personal Assistants (IPA) widely adopted in the last years. Examples of such chatbots are Apple Siri, Google Assistant, Microsoft Cortana, and Amazon Alexa.

The development of Chatbots intends to improve Human-Machine Interaction (HMI), by providing an easy-to-use interface with a natural flow of conversation. Thus, IoT management tools started to integrate chatbots because of the vast possibilities they offer for the users and the human-centric challenges tackled by these platforms.

With a Chatbot applied to the IoT, the underlying state of the system, the orchestration of devices, and respective interaction are abstracted. From the user's perspective, it only provides the most pertinent information and diminishes the learning curve to create and maintain an IoT environment. The user can use the chatbot to understand more quickly the overall functionalities of the system, whereas it also narrows the configuration challenges since a conversational approach explains more effectively step-by-step setups.

Although developments in IoT have been increasing in recent times, integrating Chatbots into IoT systems is a relatively new area. Work developed by Rohan Kar et al. in 2016, attempts to integrate both fields by detailing the key architectural components required, what are the main challenges and how to address them [KH16].

2.3 Simulation of Smart Home Environments

Intelligent Environments are environments capable of monitoring their context and the conditions of their inhabitants to enhance the overall experience within that environment [NPF+11]. The environment integrates several sensor technologies capable of data collection, depending on the specific monitoring domain and the needs of its inhabitants.

A wide range of scenarios integrates IEs to fulfill many purposes. For example, office environments for the visualization of space utilization [CP07] and the movement of employees [IWSK07]. Airports [CC03] and banks [GMBT04] also integrate IEs, intending to detect security risks from abnormal behavior. IEs deployed within the home environment are called Smart Homes (SHs), and involve the use of sensor technology to monitor the health, wellbeing, activities, and security of inhabitants over extended periods of time [DHSR08].

Several smart home environment projects follow an Ambient Intelligence (AmI) paradigm, although their focus usually is substantially different between themselves, depending on the problem dealt with [EMMN18]. Ambient Intelligence is a paradigm emerging from Artificial Intelligence,

where computers are proactive tools assisting people with their daily activities, making everyone's life more comfortable. The environments are created by building smart labs in academic settings or using lightweight hardware in a home setting. The following list includes some of the most prominent real smart homes built as testbeds:

- **Ulster University (United Kingdom)** [DNMHD09] developed a smart living environment to support the development of assistive technologies and solutions to support independent living, integrating a range of techniques for healthcare monitoring and diagnosis. This environment is in the form of a smart apartment containing a living room, dining area, kitchen, small office, a bathroom, and a bedroom, and can facilitate data collection of individual or multiple inhabitants;

- **Ubiquitous Home** [Yam07] is a smart home built to study context-aware services by providing cameras, microphones, pressure sensors, accelerometers, and other sensor devices. To provide contextual services to each resident, the Ubiquitous Home recognizes the resident by providing radio-frequency identification (RFID) tags, and using the installed cameras;

- **HomeLab** [dBAMI05] is a smart home equipped with 34 cameras distributed throughout the home space. The researcher observes and monitors the conducted experiments via an observation room. HomeLab aims to provide datasets to study human behavior in smart environments and to investigate technology adoption and usability;

- **Center for Advanced Studies in Adaptive Systems (CASAS)** [CCTK13] developed a smart home kit, easily extendable and ready-to-install in a house. The sensors embedded gather data from the residents, their activities, and environment, which is used for the generation of useful knowledge such as patterns, trends, and predictions. Research groups use the datasets collected by CASAS toolkits for their research;

- **UJAml SmartLab** [EMMN18] based on Ambient Intelligence was created at the University of Jaén to provide a smart environment where solutions in assistive technologies applied to e-healthcare could be developed. The smart lab features a fully furnished apartment with a plethora of smart sensors and actuators deployed in the apartment spaces to gather data from the human-environment interactions and fulfill the users' needs (*cf.* Figure 2.5, p. 14). It uses openHAB[4], an open-source middleware technology, to ease data management and enhance home automation capabilities.

Simulation tools can overcome some drawbacks/challenges faced by projects that envision the creation of a smart home environment. The flexibility provided allows creating a broad spectrum of scenarios similar to real life environments, abstracting away physical challenges. Such tools facilitate fast dataset generation and offer robust methods to capture the sensors' data.

The development and test of data analysis algorithms require the existence of datasets gathered from smart environments. The availability of such datasets is, however, limited because of the high

[4]https://www.openhab.org/

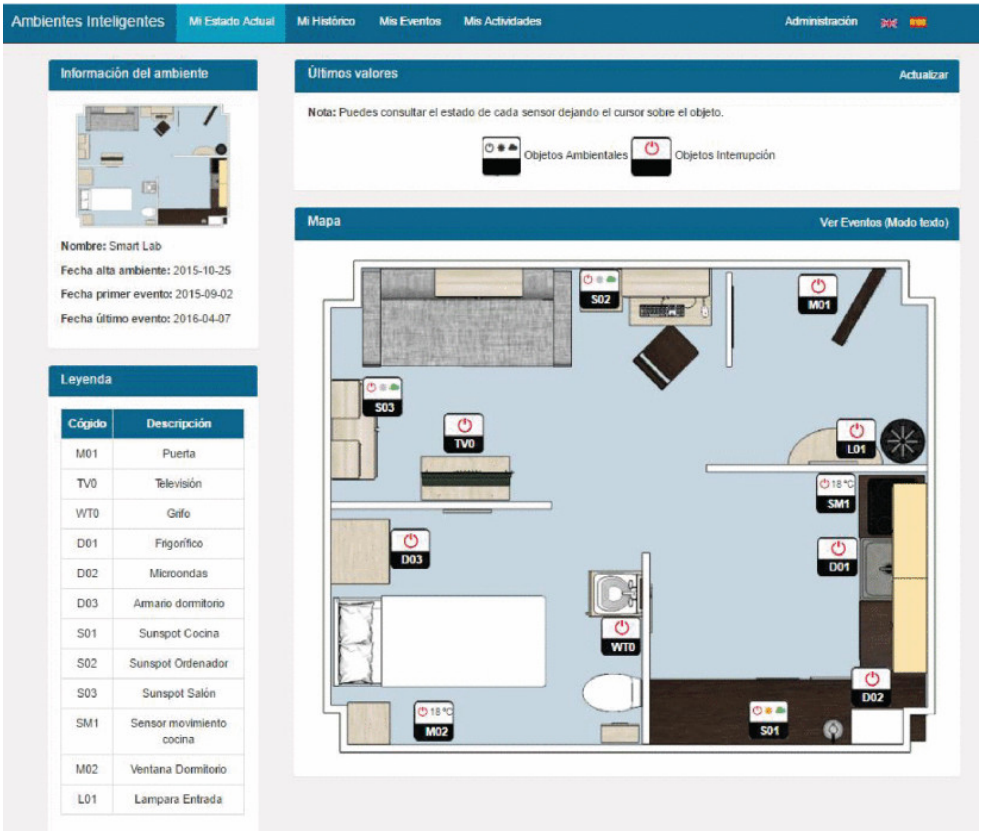


Figure 2.5: UJAml Smart Lab

costs, constraints, and limitations associated with the construction and usage of physical IEs. Using data simulation approaches has the potential to address these constraints, providing researchers with rich datasets for the testing of novel data analysis approaches, particularly at the early stages of development [SCNM14].

Synnott et al. [SCNM14] identified several challenges faced by smart home simulation research. The generation of representative datasets for a broad range of domains is a challenge that none of the simulation tools could achieve so far, whereas the focus of most simulation tools is on the visualization of smart environments with context awareness and not the generation of synthetic datasets. Another critical challenge identified is the flexibility and scalability to add new or customized types of smart devices, change their generated output, and change their positions within the smart home. On the other spectrum, the multiple inhabitants' support is also one limitation, as it is challenging to implement.

Throughout the years, IE simulation research has evolved in two directions: Model and interactive based approaches [SCNM14]. Model-based approaches require the use of activity models and can generate vast datasets representing extended periods. Typically, model-based approaches opt for the modeling of the simulation context, such as agents, activities, and processes, and as a result, the accuracy of the data generated by these approaches depends on the accuracy of the underlying activity model. Interactive based approaches provide an interface for the users to

control and capture the context in real time, which consequently increases the granularity detail of the simulation context. Interactive approaches offer the ability to record datasets with complete ad hoc control over environment and sensor simulation and may facilitate the assessment of subtle changes to environments or sensors, such as layout and sensor placement.

2.3.1 Model-Based Approach

The model-based approach uses pre-defined models of the simulation context to create the simulation flow. Usually, the models define the leading entities of the simulation, such as agents and activities. The modeling comprises typically the specification of the agent's behavior, the probability of events occurring, their influence in the environment, and the duration of each activity.

The following list summarizes the simulation tools integrating a model-based approach with the most relevance in the literature:

- **CASS** [PMHY07] is a simulation system that focuses on the context information of a smart home domain. The system responds to specific contexts through contextual rules defined by the developer and composed by a condition and an action (e.g., when the temperature is lower than 15º, turn on the AC). By using the tool, a user can detect rule conflicts and simulate rule behaviors for a smart home. As shown by Figure 2.6 (p. 16), the simulator summarizes the smart environment, the device's rules specification, and a toolbar to configure the simulation context;

- **DiaSim** [BJC09] is a simulator for pervasive computing based on actuators and sensors. The tool generates a programming framework to develop the logic for services, giving freedom for the user to develop a broad set of devices and their capabilities. It provides a GUI to define the virtual space, the custom scenarios, and to monitor the behavior of the simulated system. The tool focus lies deeply on the heterogeneity of entities that can exist in a pervasive computing system and the ability to support simulation of those entities;

- **SIMACT3D** [BABB10] is a smart home simulator specifically designed to help researchers with activity recognition problems. A set of scenarios captured from real-life experiments is integrated, which helps the recognition of activities of daily living. The tool provides a virtual smart home architecture, imbued with reliable scenarios, and allows researchers to test recognition approaches with minimal cost. SIMACT restricts the scenarios used in the simulation, as it uses pre-recorded contexts gathered from clinical trials;

- **IoT MAS** [ZKHS17] introduces an environment based agent system that simulates a smart home. The architecture includes an IoT gateway to bridge the information from the ambient environment (e.g., sensors, actuators, and house components) and the external networks. The main aim of the tool is to provide a straightforward way to manage a simulated system via remote control;

- **Kormánios et al.** [KP13] created a simulator with the purpose to model and analyze the daily activity of a single human within a virtual environment, to help the development and test of algorithms applied in ambient assisted living projects. The tool aims to model the human inhabitant, his social and physical environment so that the activities and interactions performed by an agent are meaningful and representative of real human behavior;
- **Vasilateanu et al.** [VPCG16] proposed and developed a simulator with customizable device placement and an agent-based approach to resemble human activity with a limited set of activities. The strength of this architecture lies in the agent modeling incorporated, which with the help of machine learning algorithms simulates the agents' activities. The devices recognize the activities performed and respond according to their configuration.

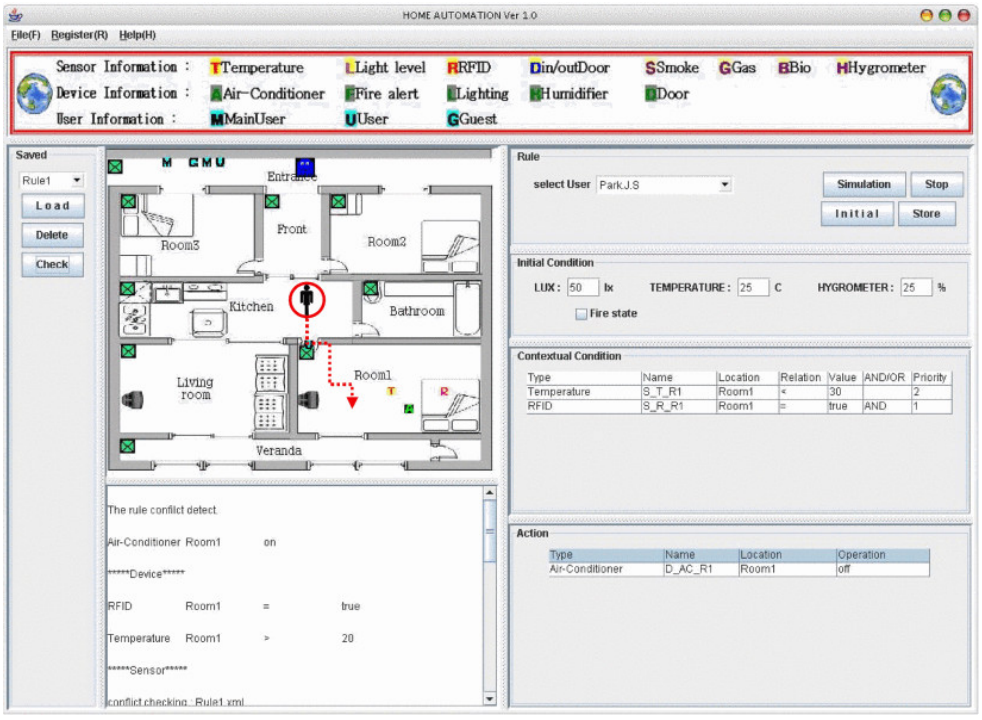


Figure 2.6: CASS Simulator

Model-based approaches can generate extensive synthetic datasets describing virtual contexts over extended periods. The quality and accuracy of the resulting datasets, however, relies heavily on the quality of the model defined. The construction of accurate activity models requires access to real test data describing the performance of the modeled activities to expand the model domain as much as possible. The approach has a significant shortcoming in representing intricate interactions or erratic/abnormal behavior in a smart home. For example, activities that have external factors or activities between various inhabitants are inherently challenging to represent.

2.3.2 Interactive-Based Approach

In contrast to the model-based approach, the interactive approach allows the users to control the agents that are part of the simulation or configure, in runtime, the simulation settings. Both actions impact the environment, whether by interacting directly with the environment, which is an active interaction or through a passive interaction, such as updating the settings.

The vast majority of tools that implement this approach incorporate a 3D view which allows the users to manage avatars and their interactions intuitively. The avatar moves and interacts with the virtual environment, which has virtual sensors and actuators. The interactions are passive or active. Passive interactions occur when an avatar indirectly performs an action that activates some condition. For example, if there is a virtual pressure sensor installed on the floor, and when the avatar walks on it, the sensor should detect this and emit a signal. Active interactions involve actions performed by the avatar, such as opening a blind or turning the air conditioner on or off [AAS+17].

The following list summarizes the simulation tools integrating an interactive approach with the most relevance in the literature:

- **UbiREAL** [NYT+06] is a ubiquitous application simulator that can simulate a 3D smart environment, devices, and how the external context of the simulation affects the devices. The simulator offers a way to control intuitively and check the communication and rules established between devices;

- **Armac et al.** [AR07] developed a tool with an interactive process to configure the environment 2D layout through a GUI. The simulator has a central gateway to connect the devices and configure their services, whereas the GUI manages their status. The goal of this tool is to test the services integrated with the devices;

- **V-PlaceSims** [LCK08] enables users to experience a smart virtual environment with context-aware information regarding activities, entities, and users. The users have an active role in the simulation, being able to explore and interact with the virtual environment, witnessing in real time the result of interactions, and how the services adapt to the current context. The tool aims to provide insights on how to better design a smart home and the ideal configuration of devices;

- **ISS** [NJD09] is a simulator that aims to help users understand the interaction between the environment, people, and pervasive devices in a smart home. The simulator architecture integrates contextual awareness, environmental effects and provides the users the possibility to interact and change the current settings. The end goal of this tool is the optimization of smart house design in terms of sensor settings and device placement;

- **SimCon** [MOG+10] extends an established building model, incorporating descriptors and parameters related to smart building technologies. The tool offers high flexibility in terms of the measured properties and simulated contexts, whereas the sensor modeling language

(sensorML) used offers interoperability. The toolset provides a runtime test environment for simulation and visualization of smart buildings, summarizing the system and its accuracy;

- **Buchmayr et al.** [BKK11] developed an extensible simulator that can generate and visualize data from a wide range of sensors and simulate faulty or unexpected behavior. This approach has the benefit of detecting poorly designed IoT systems, test the overall efficiency of the system, and produce test data;

- **Fu et al.** [QPC+11] proposed a configurable context-aware simulator for a smart home environment. Through an XML configuration, it is possible to configure the devices, their rules, and the surrounding environment. An interactive graphic interface allows the user to monitor and manage the simulation devices;

- **Ariani et al.** [ARCL13] developed a simulator capable of recognizing and responding to a person activity within a virtual smart home. The simulator gives researchers a set of tools to better understand how the sensors respond to a variety of activities, by analyzing characteristics like the number of sensors, user mobility, and line of sight. The floor plan designer integrated with the simulator is used as a validation of an initial placement configuration of sensors;

- **Intelligent Environment Simulation (IE Sim)** [SCNM14] is a tool used to generate simulated datasets that capture normal and abnormal ADLs of inhabitants. It allows the researcher to design smart homes by providing a 2D graphical top-view of the floor plan. Following an interactive approach, the users can interact with the simulation with an avatar, whereas the simulation captures ADLs;

- **OpenSHS** [AAS+17] is an open-source, cross-platform 3D smart home simulator for dataset generation, combining advantages from both interactive and model-based approaches. The replication capabilities provide a solution for generating large representative smart home datasets, overcoming the drawbacks of an interactive based approach. The tool also offers an extensive library of smart devices that facilitates the simulation of smart home environments. As seen by Figure 2.7 (p. 19), OpenSHS divides the dataset generation into three distinct phases:

  - **Design** of the initial virtual environment by building a smart home with an external designer tool and the import of smart devices;
  - **Simulation** of the context-specific events;
  - **Replication** of the initial sample dataset to obtain the extensive final dataset.

Overall, the interactive approach comes with a significant setback, as it is cumbersome to use the simulation for extended periods because the avatar completes each interaction in real-time. This approach, as highlighted by the literature, is usually used when the end goal is the visualization and testing of a smart environment context, not the generation of datasets [AAS+17].

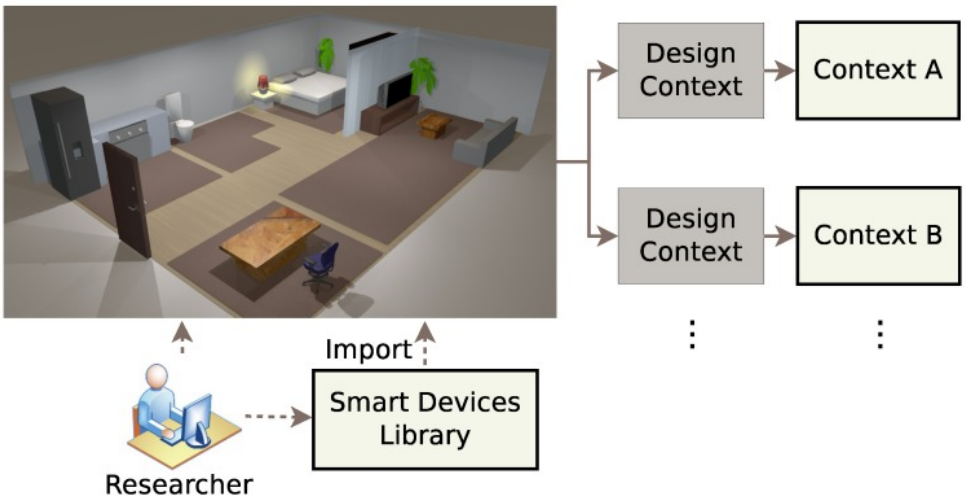


Figure 2.7: OpenSHS Simulator

2.4 Conclusions

In terms of management of IoT systems, VPPs like Node-RED continue to offer more complex functionalities and have higher management capabilities over the system, however, the required technical skills to use these platforms continues to be a barrier that few users will overcome. Chatbots are the perfect solution for the end-users because of the friendly and easy-to-use interaction they offer. These management tools are especially useful for smart environments already designed and pre-established, however, they are not ideal for situations where the design validation and testing is a requirement, or the user wants to fiddle with an idea for his smart home.

Simulators are a crucial tool for experimental scenarios, but their usefulness has a broad appeal. Because of their capability to model a wide range of scenarios, they can validate and test smart home designs using an iterative approach and reducing costs for the users. In terms of datasets generation, they can model several problem domains and create reliable synthetic datasets without the need to set up real smart environments.

The state-of-the-art is rich with efforts that focus on generating datasets for smart home applications. We can classify those efforts into two main categories; datasets generated either from real smart homes testbeds or using smart home simulation tools. Real test beds are the more reliable and used option; However, costs of setup and maintenance appear as an essential barrier for their wide adoption. Usually, these testbeds use research environments as groundwork with specific purposes in mind, so these conditions naturally limit the domain.

IoT simulation tools offer a more extensive set of features and possibilities compared to real smart home test beds. Besides offering an endless set of modeling scenarios, they offer full control over the simulation settings and parameters, so that minimal changes require zero to none effort. Simulators also have the option to define the simulation speed so that the observation is less cumbersome.

The analysis of the Table 2.2 (p. 21) shows that most of the tools featured in the smart home simulation field are not available in the public domain as an open-source project. The lack of availability of the software implementation hinders their contribution to the field, mainly to future researchers that want to develop further the efforts in the state-of-the-art or potential researchers that want to experiment the tool and verify its validity.

Regarding the integration of a rules engine, to abstract and define the interaction between devices, the number of tools that implement this is also low. The evaluation of a rules engine in this context has to fulfill two goals: First, a device's property state can depend on the state of other devices, directly or indirectly, and the communication link between the devices must be transparent to the user. Second, a device's property state can depend on the environmental context, which refers to external agents, physical properties, or time. Only four out of 19 tools implement a rules engine that complies with the goals defined for a rules engine.

The simulation tools analyzed have shortcomings in terms of flexibility. For example, for most of the tools, the set of devices used in the simulation is pre-defined, being impossible to add or customize new sensors and devices. In terms of the physical environment, none of the tools addresses the simulation of physical properties, which hinders the sensors used and gathered data.

Except for OpenSHS and IESim Extended, none of the tools follows a hybrid approach integrating a model-based and interactive solution. The hybrid tools combine the advantages of both approaches, having the ability to generate large datasets in a reasonable time interval while keeping the fine-grained interactions observed by the interactive tools. Also, fewer simulation tools focus on generating datasets or try to incorporate the visualization of a smart house and the generation of datasets.

Table 2.2: Smart Home Simulators comparison

Tool/Author(s)	Date	Open-Source	Rules Engine	Approach	Goal
IoT MAS[ZKHS17]	2018	○	○	3D Model	Visualization
OpenSHS[AAS+17]	2017	●	○	3D Hybrid	Dataset generation
Vasilateanu et al.[VPCG16]	2016	○	○	2D Model	Visualization
PerSim[LCL+15]	2015	○	○	3D Model	Dataset generation
IE Sim extended[LSJN15]	2015	○	○	2D Hybrid	Dataset generation
IE Sim[SCNM14]	2014	○	○	2D Interactive	Dataset generation
Kormánios et al.[KP13]	2013	○	○	2D Model	Visualization
Ariani et al.[ARCL13]	2013	○	○	2D Interactive	Dataset generation
Fu et al.[QPC+11]	2011	○	●	2D Interactive	Visualization
Jahromi et al.[JRM11]	2011	○	○	2D Model	Visualization
Buchmayr et al.[BKK11]	2011	○	○	2D Interactive	Dataset generation
SimCon[MOG+10]	2010	○	○	3D Interactive	Dataset generation
SIMACT[BABB10]	2010	●	○	3D Model	Visualization
ISS[NJD09]	2009	○	○	2D Interactive	Visualization
DiaSim[BJC09]	2009	○	○	2D Model	Visualization
V-PlaceSims[LCK08]	2008	○	●	3D Interactive	Visualization
Armac et.al[AR07]	2007	○	○	2D Interactive	Visualization
CASS[PMHY07]	2007	○	●	2D Model	Visualization
UbiREAL[NYT+06]	2006	○	●	3D Interactive	Visualization

Chapter 3

Problem Statement

Section 3.1 summarizes the current problems revealed by the analysis of the state-of-the-art, while Section 3.2 (p. 24) details a minimalist breakdown of the solution proposed, specifying the most relevant features. Section 3.3 (p. 24) defines the constraints applied to this dissertation. Section 3.4 (p. 25) overviews the relevant goals of the simulator and the requirements it must fulfill respectively to its context. Section 3.5 (p. 27) defines the development challenges of this simulator, while Section 3.6 (p. 28) overviews the methods used to evaluate the project.

3.1 Existing Problems

As analyzed in Chapter 2 (p. 5), the current simulators available in the literature offer several features for the simulation of smart home environments; however, from the ones available to the public domain, they have shortcomings in certain scenarios. As it stands, none of the tools simultaneously offers these features:

- **Simulation of physical properties**: integration of physical properties that are normally present in a real environment, characterized by non-binary data with diversified dynamism in terms of values;
- **Hybrid approach**: simulator with interactive configuration and control, having the fine-grained control of interactive approaches and a model-based approach for the modeling the environment context, integrating the capability to control the simulation time setting;
- **Complex devices behavior**: the devices are aware of the simulation environmental context and react to it, according to the user configuration; the device's state can depend on another device, without the specification of the underlying communication link between the respective devices;

- **Modeling extensibility**: human behavior is inherently complex and dynamic; a simulator must provide a way to model a realistic behavior of a human independently of the domain, having the flexibility to fulfill the users' needs.

3.2 Proposed Solution

The goal of this project is to develop an extensible framework capable of simulating realistic smart environments, abstracting the technical details of implementing and managing an IoT system.

The tool capabilities must offer the possibility to experiment and test realistic scenarios, without imposing a high number of constraints on the domain or wide the gap between the simulated scenario context and the real one.

The simulation tool should provide an intuitive UI for the user visualize the simulation process and easily manage all the information regarding the simulation context in real time, without having significant setbacks. The process to create a dataset must be as straightforward as possible, and it must integrate a fast-forwarding mechanism to streamline the performance of the simulation and allow the user to control the time as he intends. This feature makes the simulation less cumbersome and the gathering of datasets with long time intervals easier.

The device's behavior should be flexible enough so that the user can define complex device interactions, whether with the environmental context or between devices. The simulation tool must provide a way to simulate physical properties in a simplified but realistic way, giving the user the possibility to configure the physical properties and how the system processes the properties so that he can create scenarios with unexpected behaviors. The following example showcases the use of this simulator and its utility for the users:

A user wants to model the configuration of devices and their interactions with the surrounding environment, especially when the light of the environment changes, whether via the simulation physical world or because of devices state updates. Imagine a scenario where a person works during the night and has the lights turned off or on, depending on the environmental light that the house has. The behavior of the system follows the pattern: If the light transmitted to the light sensor is below a certain threshold, the blinds are closed, and the lights within the home turned off; Otherwise, the blinds are kept open, and the user continues his work using the environment light. Although, if it is raining outside, the blinds are closed, and the lights switch on automatically.

The system must be able to simulate realistic scenarios like the one presented above, where numerous variables affect the current state of the context.

3.3 Assumptions

During the dissertation development, and due to time constraints, some general assumptions and constraints needed to be taken into consideration.

The Internet of Things creates an ecosystem of interconnected devices and appliances that acquire, store, and share data through the Internet. This scenario raises security and privacy concerns

throughout the process chain, from the moment that the data is acquired in the device to the moment that is stored and processed in the could. Several challenges have to be tackled such as data privacy, technical authentication vulnerabilities, the human factor, and data encryption [ZCW+14], to ensure security, privacy, and safe experience for the users. The security risks associated with IoT appear as an essential research and industry problem continuously being assessed. Thus, to the scope of this dissertation, these risks are overlooked so that the authors consider the IoT system used for testing and validation purposes as secure.

As the IoT continues to be used in an extensive set of domains, the number of devices, standards, and protocols continues to increase as well. As such, the simulator engine must provide interoperability between the devices and underlying data exchanged. The data model and API used to describe the physical devices should adhere to the current standardization efforts made by the IoT community.

The simulation of real physical properties introduces a problem that requires an in-depth knowledge of physical phenomenons and has a high complexity on its own. A realistic physical simulator requires plenty of computational power and introduction of physical concepts that would increase immensely the complexity of this simulator, without addressing the problems present in the state-of-the-art explicitly. For that reason, the simulation of physics properties is simplified for the development of this simulator.

3.4 Desiderata

To ensure that the solution developed presents itself as a contribution to the ensemble of simulation tools, we describe a set of goals on the next sections, focusing on common challenges generally addressed by a simulation framework. These goals define the high-level criteria that the simulation tool must address and fulfill to consider its development a success.

As mentioned in Section 2.3 (p. 12), researchers can use machine learning algorithms with the datasets generated from simulation tools, but the reliability of such datasets is related to the underlying model. The flexibility used for simulation modeling entails the notion that the models are as reliable as the user makes them be, so the project does not intend to tackle the generation of datasets for later research use. Thus, the target audience of this project are users that intend to experiment and test virtual home scenarios to validate a real smart home design.

A project inherently has a set of forces that drive its management and development. Those forces constrain and drive the development of a project, as they define the project scope and are essential to measuring the quality of the tool. Figure 3.1 (p. 26) shows the forces that guide the development of this project.

The project has three main forces: usability, flexibility, and heterogeneity. Heterogeneity, in this context, refers to the vast set of diversified applications for IoT devices and the efforts to standardize this domain. The number of data representation models for IoT devices is increasing at

a fast rate. The flexibility of the tool supports the heterogeneity of the simulator and the capability to handle a wide variety of devices.

The modeling of realistic human behavior also requires a high level of flexibility because of the inherent complexity of human beings. The flexibility increase directly hinders the usability of the system. As the system requires an increasingly dynamic approach to represent the models, the more difficult it is to provide a user-friendly UI for the end-user.

The tool needs to offer an intuitive UI where the user can quickly understand its functionality and achieve his objectives with effectiveness and efficiency.

The forces, as evidenced above, have a direct relationship. The project development takes into account these three forces and defines its goals, trying to minimize the trade-offs between requirements.

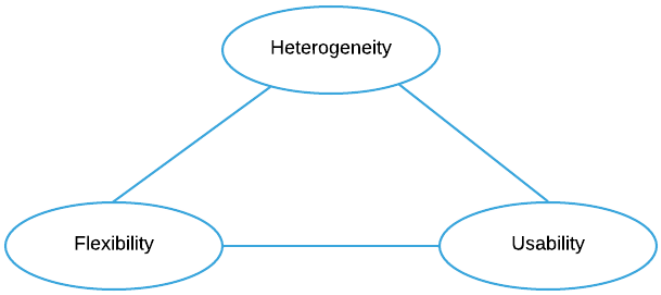


Figure 3.1: Project forces

3.4.1 Easy end-user device behavior configuration

A real smart home as a plethora of devices with different purposes and standards, so settings such as properties, actions, events, and message standards must be configurable. A proper simulator must be able to handle the heterogeneity and extend the device's library with ease to apply the simulation to a broad range of domains.

A device (sensor) is aware of the environmental conditions surrounding it. Some can sense movement and activity within the defined sensing radius, while others have complex interactions between themselves, whereas their state depends only on the state of other devices. Such approaches allow for more complex systems without the need to establish communication links.

Overall, the device behavior configuration must be transparent and intuitive for the end-user throughout the creation phase.

3.4.2 Flexible scenario customization

A smart home has an extensive set of layouts, configurations, and entities. Overall, a scenario can have a multitude of possibilities that need to be adequately represented in a virtual context.

The end-user must be able to customize the scenario he is trying to represent, independently of being a realist scenario or not. The UI must provide easy-to-use functionalities to speed up the creation of a scenario.

3.4.3 Intuitive visualization and data generation

In a real smart home, the users control the things in real time, and the physical properties of the environment are in constant change. The simulation settings in terms of devices, properties, and rules should be easily configurable while the simulation is running.

The tool must offer a view of the simulation while running and the information related to what is happening. Besides this, the generation of datasets is also a focus essential for users, which must not be a cumbersome task. It is important to gather an appropriate amount of data for posterior analysis and acquisition of insights on how to improve the practical home design, without imposing the need to wait long periods. It is vital that the user has control over the internal simulation time, with the possibility of fast-forwarding as desired.

3.4.4 Physical properties simulation

Smart homes devices are usually influenced/changed because of human activity or environmental changes (light, temperature, pressure). Some sensors depend only on external properties, so a simulator without physical properties lacks a significant component of a proper virtual smart home environment, which takes value out of the tool.

The tool should integrate the simulation of physical properties using a simplified version of a physics simulator. The end-user can configure these properties and how the surrounding entities respond to the current environmental state.

3.4.5 Easy agent behavior and activities configuration

A real-life scenario comprises a set of persons with different habits, goals, and habits. Their behaviors can be simple, like the action of sleeping or complex, such as the activity of eating lunch, which is composed of a set of sub-tasks. Also, an agent has a set of strictly personal necessities, and their fulfillment depends on the accomplishment of different activities.

The agent and activities modeling must be straightforward and easily configurable by the end-user. Because of the complexity of this modeling, the tool must be flexible, offering a model representation capable of accommodating the user requirements.

3.5 Challenges

The hypothesis of this dissertation is the modeling and experimentation of realistic scenarios within a virtual environment, applied to a subset of the Internet of Things, the Smart Home domain.

According to the desiderata described in Section 3.4 (p. 25) and the hypothesis of this dissertation, the following challenges are established:

1. "**How to express complex device behavior?**": We intend to explore the flexibility to define a wide variety of devices and their behavior. As in a real smart home, the devices respond to the activities performed by humans, to the current environment context or because of dependency with another device state. This behavior must be easily configurable by the user, abstracting the technical details of such behaviors. At the same time, we aim to understand how this flexibility impacts the usability of the end-user;
2. "**How to simulate complex user behavior?**": This question focuses on the modeling of realistic user behavior and how the data model constraints the realism that the end-user can imbue within the simulation;
3. "**How to simulate environmental properties realistically?**": Environmental properties are essential for a realistic simulation tool. However, their simulation complexity is a hindrance. We aim to explore how to create a simulator with external physical variables, opting for a simplistic yet realistic approach to the physical world;
4. "**How to integrate into a simulation tool a hybrid approach and maintain its flexibility?**": A hybrid simulation tool incorporates advantages from interactive and model-based approaches. It is important to explore how a hybrid approach can preserve its advantages while offering a flexible approach to the modeling process.

3.6 Evaluation

The authors will measure the successful implementation of this project by achieving the goals defined in Section 3.4 (p. 25) under the assumptions defined in Section 3.3 (p. 24).

These requirements are the focal point of the project and the defining forces that introduce novelty to the system. To evaluate the implementation specified in Chapter 4 (p. 29), a set of use cases will be defined, which focus on the most critical features of the project. For the use cases described, we will create one or multiple scenarios to fulfill the use case and ensure that the tool implementation was successful and effectively represents a contribution to the existent simulation tools in the literature.

The authors will evaluate the tool in terms of complexity, flexibility, and usability, and how it compares to the open-source simulation tools applied to smart homes.

3.7 Summary

This project aims to pave the way for the development of realistic simulators capable of providing an experimental environment for users to test and analyze their smart homes, as no simulation tool exists in the state-of-the-art with the requirements specified for this system. Thus, we intend to present a basis for the development of these simulators since the complexity of real scenarios is tremendous and the opportunities that arise from a simple architecture to iterate over and introduce more features are endless.

Chapter 4

Proposed Solution

This chapter describes the solution and architecture developed to tackle the problems discussed in Chapter 3 (p. 23). Section 4.1 overviews the high-level architecture of the solution, briefly explaining the main components, their relationship, and the reasons guiding the choice of technologies.

Section 4.2 (p. 32) summarizes each of the main components of the simulator, giving a full description of its features, relationship with other components, and explanation of architectural decisions. Throughout this section, the authors analyze the advantages of the solution and its shortcomings.

Section 4.3 (p. 47) explains the steps to compile the solution, the process to create a virtual smart home and its context, and run the simulation. It gives a step-by-step tutorial on how the end-user can use the solution for their purposes.

4.1 Architecture

Web-based simulation (WBS) is the invocation of computer simulation services over the World Wide Web (WWW) [BHB10]. Because of the extensive use of web services, this environment presents itself as an area of investigation for the modeling and simulation of applications, that removes the need to target different operating systems or hardware.

The framework developed adopts this architecture, integrating web technologies within the field of simulation to perform simulation experiments on the Web. The client-side technology used to build the interface is React, whereas the server-side is built with NodeJS. The technological stack integrates Typescript as it eases the development, makes the code understandable and flexible, and provides IntelliSense during development. The main shortcoming of Typescript is the extra code necessary, however, as in the authors' eyes the project can be extended in several ways, the readability and future-proof of the project are essential aspects to take into consideration.

Web-based simulation can take place either on the server side or on the client side. In the server-side simulation, the back-end server carries the simulation processing, while the client-side

displays the simulation. In the client-side simulation, the processing takes place on the client side. The framework follows a server-side simulation approach to diminish the impact on computer performance and increase the scalability.

The simulator integrates a UI built with React and provides a simple interface for the end-user to describe the virtual environment context and configure the simulation's intrinsic properties, as it is seen in Figure 4.1 and Figure 4.2. The UI developed aims to improve the overall usability of the simulator.

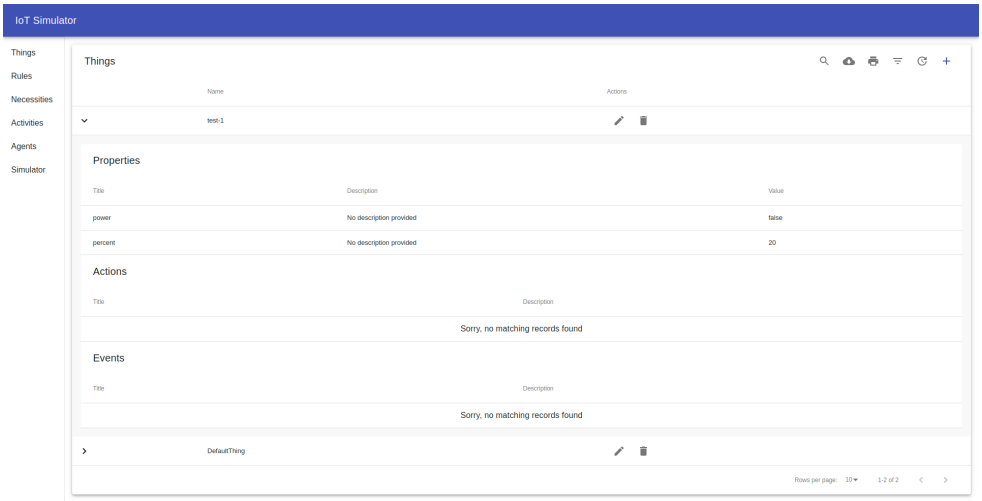


Figure 4.1: Things Management

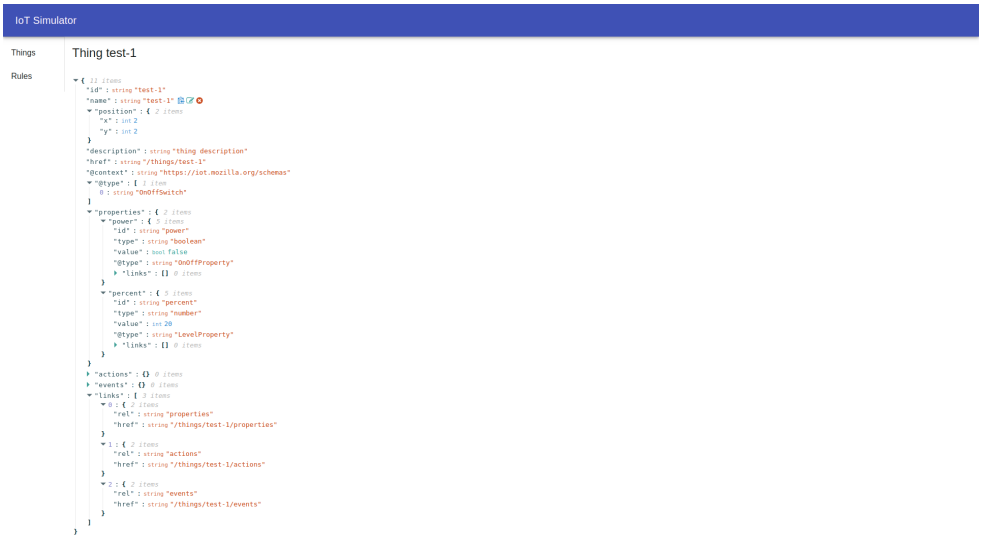


Figure 4.2: Simulation Settings

The communication between the simulator engine and the client-side is bidirectional and occurs in real time, so the framework requires a protocol to tackle this problem. The communication

protocol adopted is WebSocket, which provides full-duplex communication channels over a single TCP connection. They offer a long-lived, bidirectional communication channel between client and server with low latency and overhead. A WebSocket integration was added to visualize in real-time the simulation progress since the server can push new data whenever the simulator processes and completes an iteration.

An architecture comprising multiple services that require asynchronous communication channels between themselves, most of the times integrate a reliable and efficient message-oriented middleware. Although the framework's scope does not require such integration, as the devices can be virtualized entirely and the simulator engine manages the communication, it is crucial to develop the framework taking into account future requirements, such as the existence of simulated and physical devices.

Thus, the framework integrates a message-oriented middleware, RabbitMQ, that supports a multitude of message protocols, such as Advanced Message Queuing Protocol (AMQP), Message Queuing Telemetry Transport (MQTT) and Streaming Text Oriented Messaging Protocol (STOMP). A service connected to the middleware can be a publisher or a consumer; or both. A publisher service places messages into a queue, while consumer services subscribe to the message queue.

The use of message-oriented middleware, although increasing the technological complexity of the implementation, entails several advantages directly correlated to the characteristics of RabbitMQ. The client services connected to the middleware do not have any restrictions in terms of software specification or architecture, as RabbitMQ provides interoperability. Message queues have error handling protocols that ensure the delivery of any messages stored, with the initial configuration. Overall, the integration of this middleware allows the use of physical and simulated devices and increases the framework's scalability and reliability.

The framework architecture is composed of three main components: the Things Registry, the Rules Engine, and the Simulator Engine. The reason to adopt this architecture was the intent to create stand-alone components loosely coupled between themselves (*cf.* Figure 4.3, p. 32). Effectively, the components are independent and provide interfaces for their management and context access, which enhances their maintainability and ease of testing. The components are connected to the message-oriented middleware, so all the information that goes through RabbitMQ is accessible to each component.

Figure 4.3 (p. 32) shows a high-level overview of the simulator architecture, principal components, and subcomponents, with their respective dependencies and interfaces. The underlying data model of each subcomponent is omitted as the next sections specify all the details regarding it.

The Simulator Engine includes several sub-components directly coupled with it. The engine provides a REST API for the management of the agents, activities, and virtual physical world. The simulator engine also controls the simulation processing, the update of entities, internal timer control, and the generation of datasets.

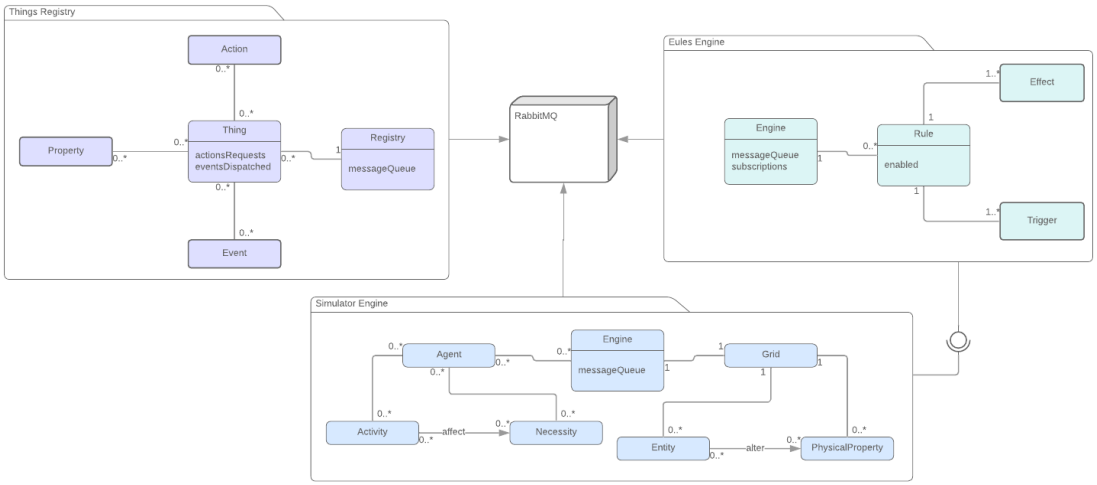


Figure 4.3: Simulator Architecture

4.2 Component Overview

This section summarizes the main components of the simulator, giving a full description of its features, relationship with other components, and explanation of architectural decisions. Each subsection overviews the implementation details, from a technical standpoint, with a description of the data model and algorithms used. On the other hand, it also describes the approach taken by the authors to solve the inherent problems that each component tackles.

The extensibility requirement stipulates the need for an approach for the users to describe the smart home context with high faithfulness. Thus, for scenarios when the logic of the simulation process depends on the context defined by the user, several components of the framework integrate complex boolean logic. This logic is defined by an array of conditions or a condition, which in turn is an object comprised of a combination of facts, operators, and values. Facts are the properties of the simulation context accessible to this logic (e.g., the simulation time or the attributes of an entity), while the operators represent the type of comparisons that the logic supports (e.g., “equal,” “notEqual,” “lessThan,” “lessThanInclusive,” “greaterThan,” and “greaterThanInclusive.”). The simulator engine compares the current state of the fact against the value and checks if the condition defined is false or not. The engine supports the conditions as described in the open-source tool json-rules-engine[1], that contains a more in-depth description of the logic integrated.

[1] https://github.com/CacheControl/json-rules-engine/blob/master/docs/rules.md#conditions

4.2.1 Things

The representation of things is a pivotal part of the framework. As mentioned in Section 3.4.1 (p. 26), it is a requirement to have a data model that offers flexibility for the end-user to describe a wide variety of devices. The industry is attempting to standardize a data model for the representation of Things; however, these attempts, so far, were in vain. Each manufacturer comes with his standard to tackle the mentioned problem, but the lack of adoption means that, in the end, the industry has another standard to add to the ever-growing list of standardization efforts. The Mozilla organization proposed a standard data model and API for the Web of Things[2] that is increasingly being adopted by the open-source community. On the one hand, the integration of this standard offers a flexible and common data model accepted by the open-source community and in the other hand presents an architecture compatible with physical things that implement the Web Things API.

The Web Thing Description provides a vocabulary for describing physical devices connected to the WWW in a machine-readable format with a default JSON encoding. The specification proposed by Mozilla was integrated and adapted to the simulator requirements, which entailed minor model changes to the specification. A thing representation, as seen in Appendix A.1 (p. 69), is modeled by the following properties:

- **type**: The type member is a description of the capabilities a device supports. The value of the type member is a string or an array of strings used by the simulator to identify if the things are sensors or actuators, and the possible effect that they have in the environment (i.e., "Emitter," "MotionSensor," "TemperatureSensor");
- **name**: The name member is a human-friendly string that describes the device. The user can configure this value to any value to identify the device within the simulation;
- **description**: The description member is a human-friendly string that describes the device and its functions;
- **href**: Represents the URL of a thing;
- **properties**: The properties member is a map of property objects that describe the attributes of the device. The property object describes an attribute of a Thing and has the following data model:
  - **type**: Primitive types such as boolean, object, array, number, integer or string;
  - **name**: A string providing a human-friendly name;
  - **description**: A string providing a human-friendly description;
  - **metadata**: Group of optional metadata that describe the property and allow the flexible description of a property object. This metadata includes the unit specification, an enumeration of values for the property, a boolean showing if the property is read-only, minimum and maximum values, and a number specifying if the value must be multiple.

[2] https://iot.mozilla.org/wot/

- **actions**: The actions member is a map of Action objects that describe device usage more complex than the simple setting of a property, which sometimes is insufficient. For example, an action can describe a function which takes a period to complete, manipulates multiple properties, or has a different outcome based on provided arguments or current state. An Action has the following data model:
  - **title**: A string providing a human-friendly name;
  - **description**: A string providing a human-friendly description;
  - **input**: Group of optional parameters that thoroughly describe the action. The parameters can include a primitive type (boolean, object, array, number, integer or string), a unit, and minimum and maximum values.
- **events**: The events member is a map of Event objects that define the events a device might emit. Events may be used to monitor changes in multiple properties or events that describe something other than a property change, such as the reboot of a device. An Event has the following data model:
  - **title**: A string providing a human-friendly name;
  - **description**: A string providing a human-friendly description.

The things data model follows to the extent possible the API described by the Web of Things API, integrating with the context of the simulation, such as the time dimension. The device registry acts as a gateway and point of management for the things using a REST API. Regarding asynchronous communication with the messaging middleware, RabbitMQ, each thing maintains an open connection to it, publishing properties, actions, and events to its topic of reference and listening for external interactions.

The decoupling between the things registry and the simulator engine supports the introduction of simulated and physical devices.

4.2.2 Rules Engine

The simulator should integrate a component to model the automation and complex device behavior existent in a smart home environment. In terms of depth of functionality, it must support building complex logic, handle the time dimension within the simulator, and deal seamlessly with uncertainty. It also should be flexible and extensible for the end-user.

A rules engine is a software component that enables developers to model the system's behavior in a declarative way, which is the ideal solution for the requirements mentioned above. Essentially, a rules engine is a system that applies a series of decisions to determine how to react to a particular situation or what to do with a piece of data. Rules engines are powerful automation machines and typically address different problems. Forward Chaining Engines, Condition-Action engines, Flow Processing[3] engines are examples of the different rules engines that exist.

[3] https://www.waylay.io/rules_benchmark/

The evaluation of rules technologies is carried out by analyzing its depth of functionality, complexity, and adaptability. As one requirement of the simulator is to provide complex device behavior but at the same time maintain a suitable level of usability for the end-user, the rule engine implemented follows a Condition-Action approach, which is a popular engine and easily configurable.

A rule is simply a trigger, an effect, and an enabled condition. Triggers are conditions that need to be fulfilled so that the effect, a change on a thing, is carried out. Both triggers and effects rely on the awareness of the context and the state of the virtual home devices. In short, the statement "if trigger happens then effect occurs" represents the concept of a rule. Triggers can be anything from observations of things like "kitchen light is on" to abstract statements like "it is 9:00 AM". Effects are simple actions performed on things, such as "turn the light on" or "set thermostat to 25º".

The Rules Engine manages the rules integrated into the simulator, providing a REST API with the possibility to create, update, and delete rules. This component has a connection link with the message-middleware and coordinates the rules by listening to the update of things properties, as it is shown in Figure 4.4. It maintains a list of subscriptions directly related to the things and listens to the property status, checking if the property update activates any trigger. If a trigger is active, the rules engine forwards the effect to the middleware.

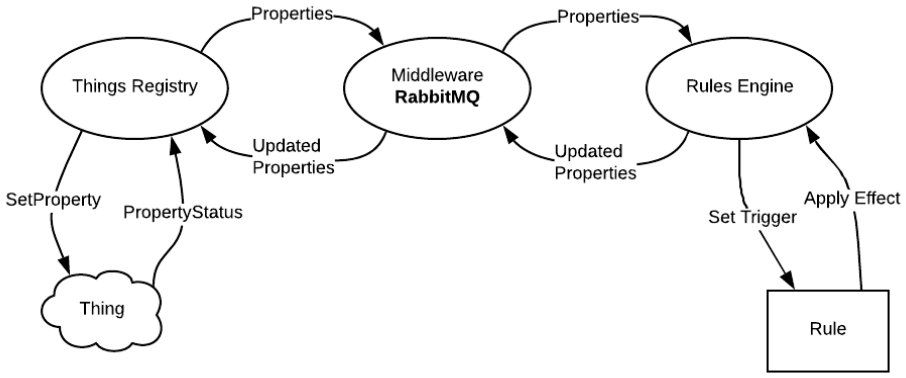


Figure 4.4: Rule Engine process

As the engine needs to keep track of the state changes and inform the dependent rules of the change, the **observer design pattern** is particularly useful for this situation. The engine maintains a list of all the subscriptions, called observers, with the information of the rules and properties that they depend on, and notifies them automatically of any state change.

4.2.2.1 Triggers

The rules engine implemented includes several triggers with different levels of complexity:

- **Property:** Observes the value of a specified thing property. Represents an abstract class;

- **Boolean**: Extends the property trigger, which is activated when the value matches the boolean defined;
- **Equality**: Extends the property trigger, which is activated when the value is equal to the one specified in the trigger;
- **Event**: Activates when occurs a thing emits the specified event;
- **Level**: Activates when a numerical property is less, equal to or greater than a specified value;
- **Multiple**: Defines a set of triggers. It is possible to define the boolean logic that commands the trigger activation, i.e., AND or OR;
- **Schedule**: Trigger with contextual awareness regarding the environment that occurs in a specific schedule defined through a cron expression[4];
- **Time**: Trigger with a condition directly associated with the time context of the simulation.

4.2.2.2 Effects

The rules engine implemented includes several effects with different levels of complexity:

- **Action:** Effect that describes an action to perform on a thing;
- **Multiple:** Set of effects;
- **Property:** Abstract class related to a property of a thing;
- **Pulse:** Extends the property action. Temporarily sets the target property to a value before restoring its original value (when the trigger is no longer active);
- **Set:** Sets the target property to a value.

The user specifies the rules through a JSON file, as seen in Appendix A.2 (p. 70), or via the UI. Since the number of triggers and effects is relatively large, the **factory design pattern** is useful to deal with the creation of such objects. A TriggerFactory and EffectFactory provide a standard interface to create the correct subclass of the object, determined by the user specification.

4.2.3 Agents

To simulate a realistic context, it is not just necessary to describe the virtual environment but the inhabitants with complex behavior that interact with the smart house. The representation of these inhabitants must be as similar as possible with a real person. The agent definition must be flexible, with the capability to define interactions with the house's appliances and devices, whereas the sensors should be able to recognize the agent's activities and behaviors inside the environment.

[4] https://www.baeldung.com/cron-expressions

Usually, a person acts on a need. Real-life simulators like Sims use this approach to represent a human being, where the agents if no explicit action has been given, will perform the activity that fulfills their needs. For example, let us assume that Tiago is an agent inside the system. If Tiago is exhausted, he will go to bed and sleep to replenish his energy. From the example given, it is easy to identify sleep as the activity and energy as the need. A set of activities fulfills one or more necessities, and a necessity fulfillment decreases throughout time.

The agent modeling integrated into the framework follows an approach closely related to the one used by Sims. Each agent has a set of necessities to which it gives priority. The activities that an agent performs depends on its necessities and have a direct impact on it. Each necessity decays throughout time, so the agents usually have to realize activities to fulfill necessities, just like in real life. The choice of the activity to perform next depends on two factors: the current fulfillment of that necessity and the specified priority.

An agent, to perform a set of activities, has to move around the smart environment. The A* search algorithm is used to determine the path that the agents traverse within the virtual environment, defined as a 2D grid, as illustrated in Section 4.2.5 (p. 41). The algorithm takes into account the paths, the cost involved to "walk" from the current position to the goal position, and the obstacles encountered in the virtual home. Besides being one of the most efficient algorithms for such scenarios, it guarantees to find the optimal path if it exists. The agent data model has the following properties:

- **name:** String providing a human-friendly identifier for the agent;
- **position:** Identifies the agent position within the simulation environment. The simulator engine checks the validity of each agent when the simulation is initialized;
- **icon:** String specifying the Unicode character that defines the agent's visual representation. The framework only accepts Unicode characters associated with a FontAwesome icon[5];
- **color:** Color of the agent in the simulation UI;
- **activities:** List of activities that the agent can perform. Section 4.2.4 (p. 38) describes the Activity data model;
- **necessities:** List of necessities representing the needs that the agent wants to fulfill. The following properties define the Necessity data model:
  - **name:** String identifier for the necessity;
  - **priority:** Defines the priority given by the agent to the particular necessity. The value ranges from 1, the highest priority to 5, lowest priority;
  - **fulfilment:** Defines the necessity current fulfilment. The value ranges from 1 to 100, decreasing with the simulation time and is affected by the activities performed by the agent;

[5] https://fontawesome.com/icons

  - **decay:** Corresponds to the necessity decay throughout time. This property is flexible and has several configurations. The object has a time unit, i.e., S, M, H, and D that correspond respectively to the decay per second, minute, hour, and day. Regarding the decay value, it can be a number, a range, or a rule. A number represents the simplest case and is a constant decrease per time unit. Range defines a minimum and maximum value that the fulfillment decreases per time unit, whereas the final value is a random value in that interval. The rule represents a complex behavior, where the decay value depends on the simulation context.

Appendix A.3 (p. 72) exemplifies the representation of an agent. Both the necessities and activities specified in the agent JSON are just the respective identifier, since the simulator engine, when the simulation is running, provides access to the instances of both necessities and activities. Besides the agent representation, there are examples for the types of necessities that the simulator supports.

The necessity "hygiene" with decay type "NUMBER" is the simplest one, as the necessity fulfillment decays exactly 1 unit per hour elapsed. The necessity "comfort" with decay type "RANGE" implies that the necessity fulfillment decays a value between 5 and 10 per minute. The necessity "hunger" with decay type "RULE" has a more complex logic that depends on the current simulation time. If the simulation hour is greater than 12, the necessity fulfillment decays 1 unit per hour; if the simulation hour is equal or less than 12, the necessity fulfillment decays 0.5 units per hour.

4.2.4 Activities

The necessities and activities are an integral part of an agent's behavior, as in a real-life scenario, an inhabitant performs activities to satisfy needs. Usually, a series of actions compose an activity performed by an inhabitant, intending to reach a specific goal. For example, if a person wants to sleep, it will first close the curtains, go to the bathroom, then to the bedroom, and only after that will sleep. As it is clear, an activity is a sequence of sub-activities, so the activities modeling must support the definition of complex activities.

As activities are usually related to the home appliances or devices and respective interaction with such objects, to perform an activity, the surrounding environment must satisfy a set of preconditions. When the preconditions are satisfied, the agent performs the activity, which results in a set of postconditions. An activity is defined as a set of tasks, with preconditions to be fulfilled before the beginning of the activity and postconditions that will occur at the end of the activity.

Appendix A.4 (p. 74) shows the JSON representation of an Activity, which is described by the following data model:

- **name:** Activity identifier;
- **duration:** Total duration of the activity in seconds, minutes, hours or days, with the specification of the value as a number or a range;

- **group:** Optional parameter that specifies the entity group where the activity is performed. The simulator engine uses this property to analyze and choose the entity where the agent will perform the activity. Section 4.2.5.2 (p. 44) explains the entity group property;
- **effects:** List of effects that the activity has on the agent's necessities. After the activity completion, the fulfillment of each necessity is updated with the respective effect;
- **preconditions:** List of preconditions that must be true before performing the activity;
- **postconditions:** List of postconditions resulting from the act of performing the activity;
- **dependencies:** List of subtasks that are part of the activity.

In terms of customization, a condition represents a flexible object with a property, a value, and a data type. Since the conditions are associated with the surrounding entities, the property parameter refers to a property of an entity, and the value property specifies the value of that property after the condition is met. For example, let us assume the activity of getting food from a fridge: The precondition is that the fridge is open and the postcondition is that the fridge is closed. The following annotation represents the preconditions and postconditions:

```
1 {
2   preconditions: [{ property: "open", value: true, type: "BOOLEAN" }
      ],
3   postconditions: [{ property: "open", value: false, type: "BOOLEAN"
      }],
4 };
```

Listing 4.1: Precondition and postcondition representation

The preconditions and postconditions of an activity can be configured with complex logic incorporating the specification of facts regarding the time context of the simulation.

The agent maintains an internal queue with the activities to perform. This queue is especially useful due to the existence of complex activities with multiple subtasks. As mentioned before, the agent selects the activity and respective subtasks that have the most significant effect on the necessity with the lowest fulfillment. After the activity selection, the simulator checks if it has subtasks, performing them first; Otherwise, the agent enters the walking state, checking the position to perform it. If the position is available, the agent moves to that position; If not, the agent does not complete the activity.

The act of performing the activity per se is composed of three parts: Interact with the environment and carry out the preconditions, complete the activity with the specified duration and carry out the postconditions, once again interacting with the environment. Figure 4.5 (p. 40) shows the flow implemented for an agent to perform an activity.

As the agent needs to alter its behavior throughout an activity, the simulator integrates the **state design pattern** for this specific situation. The state pattern is a behavioral software design pattern

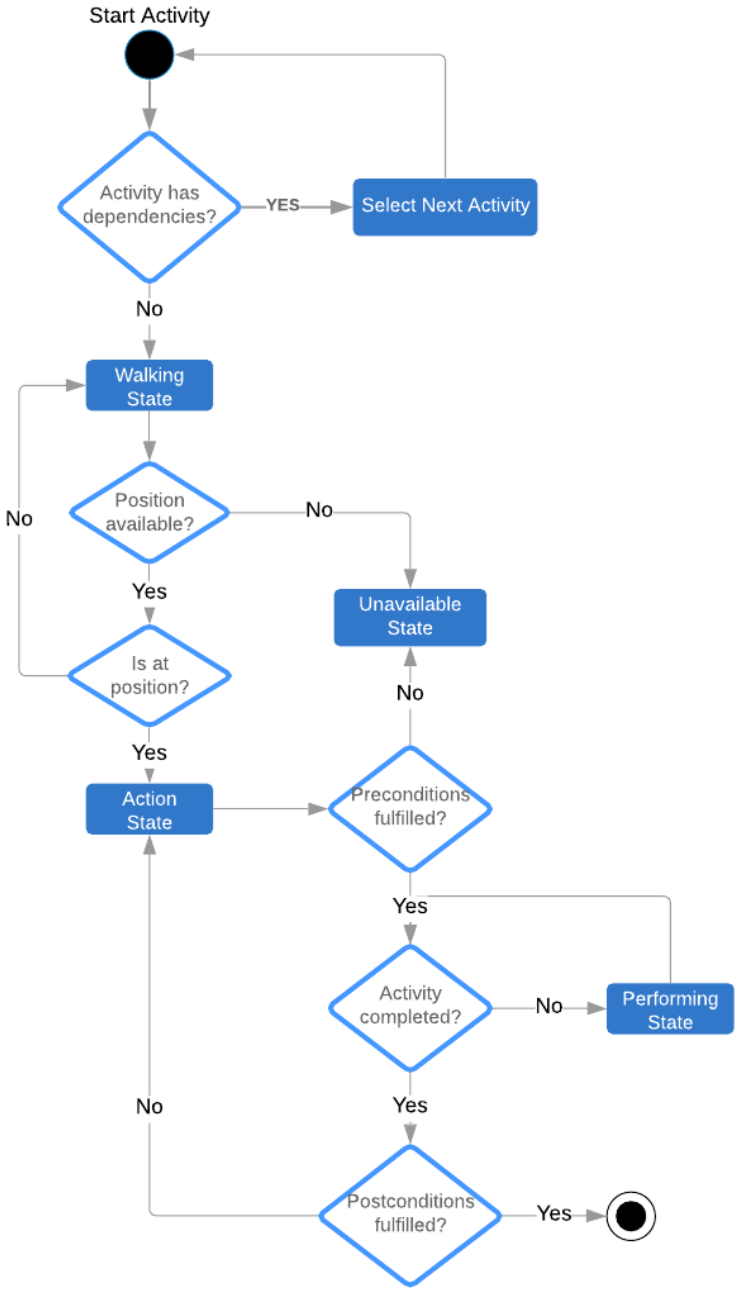


Figure 4.5: Activities Flow Diagram

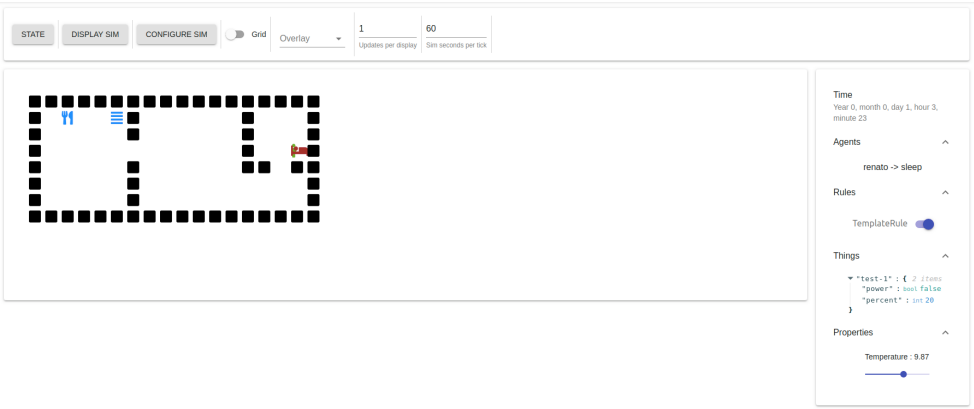


Figure 4.6: Simulation View

used to encapsulate different behavior for the same object, based on its internal state. The pattern is a cleaner way for an agent object to change its behavior at runtime without resorting to conditional statements and thus improve maintainability.

4.2.5 Virtual Environment

The simulation tools for smart homes in the state-of-the-art usually provide a customized editor in 2D or 3D so that the users can design their virtual smart homes. Undoubtedly, a 3D editor and viewer is more appealing and user-friendly but brings along a disadvantage that discourages users from frequently using the tool. A smart home has an extensive set of entities and intricacies, so the design of a smart home in a 3D editor is cumbersome and sometimes involves the use of external tools and subsequent know-how to do it, which hinders, even more, the use of a 3D editor. OpenSHS is an example of a 3D simulation tool that requires experience with Blender to create the virtual environment.

A 2D display reduces the layout complexity but keeps an acceptable degree of reliability and realism, which is ideal for the scope of this framework. For example, the simulation of physical properties in a 3D environment would be extremely complicated; however, it would not be an essential feature nor bring significant improvements to the framework. Thus, the framework represents the smart home environment with a 2D rectangular grid. The central section in Figure 4.6 shows a UI representation of the virtual smart home.

The framework has to simulate physical phenomena, which is a complex area of research that requires vast knowledge in terms of physics and mathematics. The accurate representation of a physical model is out of the scope of this project, as a simplistic yet realistic simulation of physical conditions is acceptable for it.

After some analysis how on to simulate such complex phenomena, Cellular Automaton (CA) appeared as one solution that could tackle this problem, since these systems can simulate a variety of complex real-world systems, from a simple configuration. CA are dynamic systems where space, time, and variables are discrete. They are useful for faithful parallel processing of

lattice models, exhibiting, and illustrating physical concepts. They constitute computable models for complex phenomena and large-scale correlations that result from very simple short-range interactions [Vic84].

A CA is a model of a system of "cell" objects, each one a stateful automaton, with the following components:

- **Cellular space:** A collection of cells arranged into a discrete lattice, such as a 2D grid;
- **Cell state:** The information representing the current condition of a cell;
- **Initial conditions:** What state the cells are in at the start of the simulation (simulation time t=0);
- **Neighborhood:** The set of adjacent/nearby cells that can directly influence the next state of a cell. The neighborhood can be defined in multiple ways, depending on the cellular space dimension.

A step in a CA (advance the simulation time by one unit) generates a new global state. Usually, there is a constant rule that determines the new state of each cell, taking into account the current state of the cell and the states of the neighboring cells. The rule is applied to the whole grid in parallel or synchronously.

In a 2D grid, there are two types of neighborhoods: von Neumann neighborhood and Moore neighborhood (*cf.* Figure 4.7). Assuming a cell C, von Neumann neighborhood is the smallest symmetric 2D aligned neighborhood usually described by directions on the compass $N = \{$N, W, C, E, S$\}$. Formally, the von Neumann neighborhood is the set of neighbors $N = \{0,-1\},\{-1,0\},\{0,0\},\{+1,0\},\{0,+1\}$

Moore neighborhood is a simple square with the cell C in the center. Usually, cells in the neighborhood are described by directions on the compass $N = \{$NW, N, NE, W, C, E, SW, S, SE$\}$. Formally, the Moore neighborhood is the set of neighbors $N = \{-1,-1\},\{0,-1\},\{1,-1\},\{-1,0\},\{0,0\},\{+1,0\},\{-1,+1\},\{0,+1\},\{1,+1\}$

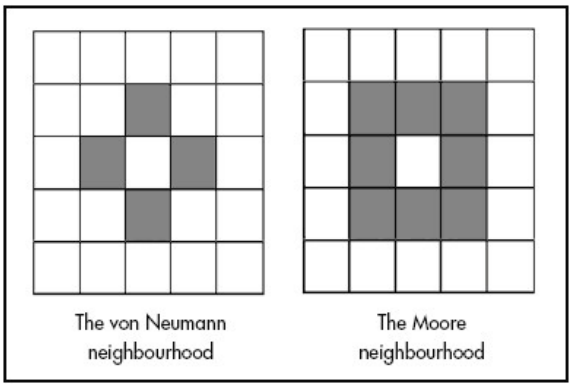


Figure 4.7: 2D Grid Neighborhoods

The framework integrates a CA to simulate physical phenomena within the virtual environment. Thus, the user has the option to configure the physical properties of the world. The goal of this action is to give freedom and flexibility for the user to simulate scenarios that can have extreme conditions or erratic behaviors in terms of physical context.

With the notion of a CA in mind, the integration of physical properties has two phases: First, the cellular space has the definition of the properties metadata, such as the title, description, SI unit and maximum and minimum limits for the property value within the simulation. Second, the definition of the absorption and diffusion rate of a given cell for a physical property, acknowledged as the rule to define the new cell state.

The physical properties can have different behaviors depending on the object that they interact. A smart home has a multitude of home appliances, connected or not to a thing. For example, a smart home can have a group of doors, tables, and beds with no internet connectivity and a group of appliances such as blinds, lights and temperature sensors with interconnectivity between them and the internet. These home appliances, in the simulator context, are denominated entities, which influence the flow of the physical environment.

In terms of implementation, the cellular space, a 2D grid, has a set of metadata describing the physical properties represented in the virtual world. Each cell of the cellular space has a defined neighborhood and is associated with an entity, holding the actual conditions of that cell. The entity's absorption and diffusion rates influence the next state and the state of the neighboring cells. A rate has a range between the interval [0, 1], e.g., An entity with an absorption rate equal to 0 does not absorb any property emitted by the neighbors, while a diffusion rate equal to 1 means that the property value is completely diffused to neighboring cells.

4.2.5.1 Properties

The simulation world has a set of physical properties described by the user. The user can describe the context of the virtual physical scenario as he intends, with the physical property data model defined for the framework. Appendix A.5 (p. 74) shows examples of physical properties supported by the simulator, which have the following data model:

- **title:** String identifier for the physical property, such as temperature, light, pressure;
- **description:** Optional parameter that describes in a human-friendly way the physical property;
- **unit:** Specification of the SI unit of the property;
- **limits:** Range parameter that defines the minimum and maximum values the property can take within the environment;
- **absorption rate:** Defines the rate at which a cell absorbs the amount of a property emitted from the neighboring cells. The specification of this rate is flexible and thoroughly customizable by the user:

  - **type:** Specification of the rate type, which can be a number or a rule;
  - **value:** The value parameter can be a number or a rule, as mentioned above, although the value range must be between 0 and 1. The number represents a constant absorption per time unit defined. The rule represents a complex case of a rate. Let us assume a blind that can be open or closed and the absorption of light. The absorption of this entity depends on its current state. If the blind is open, the absorption has a higher value; Otherwise, the value is 0;
  - **unit:** Specification of the time unit used for the rate, which can be one of S, M, H, D. This parameter defines the unit of time needed to absorb the value received.

- **diffusion rate:** Defines the rate at which a cell emits the property to the neighboring cells. The specification is similar to the one defined for the absorption rate.

4.2.5.2 Entities

An entity in the simulator context, as mentioned before, can represent the home appliances of a house. However, a smart home is not designed only with appliances. It is necessary to represent static objects, empty spaces, or even the edges of the simulation. In the simulator context, all the elements, regardless of their nature, are represented as an entity. There are three types of entities: "Edge," "Static," and "Interactive."

The following properties represent the entity data model:

- **position:** Definition of the entity position within the virtual smart home. The simulator validates this position, making sure there are no collisions with agents or other entities;
- **walkable:** Boolean parameter that defines if the agent can walk over the entity; For example, assuming the entity is a door, it is walkable if it is open. The user can customize entities and define these situations by specifying the state dependency that the walkable property depends on;
- **properties:** An entity has the specification of the diffusion rate and absorption rate of the physical properties defined in the world;
- **state:** Map of entity attributes characteristic of the home appliance. An entity property is an attribute of an entity and has the same data model as a thing property;
- **type:** Represents a list of common characteristics or a group that the entity belongs. For example, a lamp would have the types: "Emitter" and "Lights." The type is used for the agents to lookup the entities where they can perform certain activities and identify the behavior of the entity within the environment.
- **color:** String representing the entity color in the 2D viewer;

- **icon:** String representing the entity icon in the 2D viewer. The framework only accepts Unicode characters associated with a FontAwesome icon[6];
- **available:** Boolean property that states if an agent is using the entity.

In terms of configuration, the user can define the visual representation of entities and their effect on physical properties. Since it would be cumbersome to define all the entities, the user can define a common edge and static entity, and the simulator shares the configuration with the remaining entities of the same type.

The delimiters of the simulation space are edge entities. These entities have slightly different characteristics compared to the remaining entities. First, the edges are the outermost cells of the grid. Second, the edge cells define the initial conditions of the environment, as they are the source of those conditions, which implies an inward flow of the physical conditions. Finally, the absorption function of an edge entity does not update the value in that cell; The value is absorbed from the neighboring cells but does not affect future calculations, which maintains a consistent set of environmental values in the edges.

Static entities represent static objects in the simulation with whom the agents cannot interact, e.g., walls and space. These entities do not have attributes associated with since the entity state is constant throughout the simulation process. In contrast, the entities have a set of physical diffusion and absorption for each property, so, although having a passive role in terms of interaction with the agents, these entities influence the conditions of the environment. To ease the design of this entity, the user can define a default static entity, replicated to all the cells that do not have a customized entity. In the presence of several static entities with the same configuration but different positions, it is possible to define the entity with the specification of the position as an array (*cf.* Appendix A.6 (p. 76)).

Interactive entities represent objects on which the agents can perform activities and interact. The entity has a group of attributes that change throughout the simulation, whether by an active or passive interaction. The preconditions and postconditions of an activity are directly linked to the state of an entity. An entity associated with a device has the list of entity attributes automatically synchronized, with bidirectional updates managed by the simulator engine. The simulator engine listens to the message middleware for any device change that occurs and notifies the entity related to the particular device. If an agent interacts with an entity and alters the state of an entity, the engine propagates the change to the message-middleware.

Regarding the impact of interactive entities in the physical conditions, the user has the means to define the absorption and diffusion with complex boolean logic depending on the state of the entity.

4.2.6 Simulator

The simulator engine is the core of the framework. The engine provides a REST API to manage all the data that is part of the simulation context and integrates a WebSocket server to maintain

[6] https://fontawesome.com/icons

bidirectional communication with the clients. Since the engine is such a crucial class, it uses the **singleton design pattern** to ensure that the class has only one instance and there is a global point of access to the instance.

The update procedure of the simulator (*cf.* Appendix B (p. 79)) is continuously called when the simulation is running. The function first checks the simulation state, and if there are no state synchronization issues. The next step verifies the clients connected to the WebSocket server and if the update counter is equal to the update rate defined by the user, which is a condition necessary for the engine to send the simulation data. In short, the update rate, a value in the range [1, 60], specifies how many updates are sent to the client-side application. After the client update, the simulation data, associated with the current timestamp, is then written to the dataset file and sequentially updated for the new timestamp.

When the user initializes the simulation, the engine uses the **memento design pattern** to save the initial configuration. Also, it is verified the validity of the context created to ensure that the simulation runs smoothly since it would not make sense to simulate with invalid positions. The initial configuration is restored once again when the user changes the simulation state to finished, allowing him to continue configuring the simulation context and adapt it to his requirements.

A simulation model has an underlying time model which can be discrete or continuous time. Choosing a discrete time model means that time is measured in steps (with equal durations), and all temporal random variables used in the model need to be discrete (i.e., based on discrete probability distributions). A continuous time model entails the need to define a simulation time granularity (the time delay until the next moment). The simulator supports a discrete time model with the step duration configurable by the user with the restriction of being in the interval [1, 60] seconds.

An essential issue in simulation is the time progression specification. Usually, simulators support fixed-increment time progression, next-event time progression, or a hybrid approach of both cases, e.g., a social science model based on "ticks" and specific events. A model with clear fixed-increment time progression defines an update procedure and a time increment parameter, but no event types, while a model with next-event time progression defines event types and event rules, but no time increment parameter. The simulator embeds the fixed-increment time progression approach since it is ideal for modeling and measuring the evolution of the physical conditions. If the simulator did not support the simulation of physical conditions, the use of the next-event time progression would be acceptable.

The generation of datasets has the intent to capture the essential data of the simulation scenario, with the highest usefulness for posterior analysis and applicability in machine learning techniques. Thus, the simulator creates two dataset files with the format **<initial_timestamp>-agents.txt** and **<initial_timestamp>-properties.txt**, using the tab character, ‘\t,‘ as the delimiter of the data recorded:

- The agent’s file contains information regarding the activity of each agent. The header of the file represents the dataset metadata, specifically the identifier of each agent, separated by the tab character. For each timestamp recorded, the file presents the activity that each agent was performing at the recorded moment;

- The properties file contains information regarding the active rules and the state of the devices. The header of the file is composed of the pattern "timestamp rules <thing_id>:<property_id>". Each iteration recorded in the dataset shows the set of rules active and the value of each device attribute.

The project's scope has the intent to simulate physical properties. However, the generation of the physical properties values, in the authors' opinion, lies outside of it. Thus, to have a truthful and reliable source of values for physical properties, the simulator uses a weather dataset with information about the weather conditions throughout one year. This dataset has information about temperature, humidity, wind speed, visibility, luminosity, and pressure, which are, by default, integrated within the virtual environment. Nevertheless, the user can define new physical properties, yet their source value must be manually updated.

4.3 How To Use the Simulator

The application was built on top of Docker. Docker is a manager for infrastructure that includes the capacity to build and run applications on top of containers[7]. We built our application with containers because it allows us to have an essential requirement: Reproducibility. A container is guaranteed to be identical on any system that can run Docker, which allows the deployment and use of the simulator on any target software. Another benefit that we took advantage of was isolation and consequent decoupling of components because of the natural separation of containers.

To use the simulator, it is necessary to have Docker installed. With it, the simulator is initialized with the command 'docker-compose up', which will install all the project dependencies and run the application. The application UI is available at localhost:3000. The user can configure the simulation context via the UI or directly on the JSON files existent on the application root folder.

The UI, on the simulator tab, shows the view of the simulation. The page is divided into three sections: menu, viewer, and sidebar (*cf.* Figure 4.6, p. 41). The menu contains general settings to control the simulation state and display. In terms of control of the simulation, the user can start, pause, and finish it or configure the update rate and simulation speed. Regarding the viewer, it is possible to toggle the display of the cells delimiter and the display of an overlay of physical properties reflecting the environment conditions of the simulation.

The viewer displays the simulation representation and how it evolves when running. To ease the understanding of the simulation context, the entities in the viewer display their current physical properties and state, while the sidebar showcases the agents and their current activity, operating rules, the state of the devices and the source values used for the physical properties. When the simulation is paused, the user can interact and change the highlighted context information via the sidebar.

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

4.4 Conclusions

The framework incorporates several components that make it a great simulation tool with a high degree of customization for the end-users. The architecture of the framework was developed having in mind future requirements for it and the intent to be considered a stepping stone for simulation tools. The project developed is available for the open source community with the intent to foster the availability of simulation tools alike[8].

Chapter 5 (p. 49) describes how the authors evaluate the simulator and its functionalities, as it is essential to analyze how well the simulator works and how it delivers its proposed features.

[8] https://github.com/renatoabreu11/iot-simulator

Chapter 5

Evaluation

Because of the framework inherent characteristics, it is challenging to come up with coherent and reliable evaluation methods that can uphold the advantages of this simulator to the scientific community.

The evaluation method relies on the creation of custom scenarios and the analysis of the simulation process using the scenarios created. For each custom scenario, the authors intend to achieve a defined set of goals that go towards the attainment of the desiderata described in Section 3.4 (p. 25).

5.1 Use Cases

To simplify the evaluation process, let us assume a common scenario to use throughout it: A researcher intends to create a sophisticated smart space in his own house to add more comfort for himself and his family. More than creating a pervasive computing system, he aims to simulate the real conditions present in his daily life and iterate over the system design, through the analysis of the data produced and the simulation settings. The configuration of the scenarios described in the next sections is present in Appendix C (p. 81).

5.1.1 Use Case 1: configurable and extendable device behavior

The researcher wants to add a set of devices, i.e., sensors and actuators, to the simulation with a set of different characteristics that interact with the surrounding environment. It is also important to configure the properties of the devices to have higher control over the data gathered. The following list shows the devices that the researcher wants to integrate within the simulation:

- Electric heater - Emitter;
- Motion sensor - Sensing;

- Temperature sensor - Measure.

The devices exemplified have different behaviors within the virtual environment, having the capability to change the environmental temperature, sense movement, and measure the environmental temperature.

- **Simulation details:** Section C.1 (p. 81) contains the partial representation of the context created for this scenario. There are three devices associated with entities: An electric heater that warms up the environment; A motion sensor that detects movement; A temperature sensor that measures the environmental temperature;

- **Expected behavior:** The electric heater emits heat to the surrounding cells, as the surrounding cells quickly increase their temperature compared to the remaining ones; the sensor measures the temperature and synchronizes the information with the device; the motion sensor detects movement when an agent is close to the sensor;

- **Analysis:** The entities effectively work as expected, with a seamless synchronization between entities and devices (*cf.* Figure 5.1). The simulator provides a data model with an extendable device behavior; however, the main shortcoming of the configuration is the need to define attributes and properties with the same identifier when they are related. For example, a device that measures the temperature requires an attribute with the same identifier as the physical property defined. The use case addresses the desiderata related to the easy configuration of devices specified in Section 3.4.1 (p. 26).

timestamp	rules	electric-heater:power		electric-heater:temp		motion-sensor:power	motion-sensor:motion	temperature-sensor:power	temperature-sensor:temp
2019-01-01T00:00:00.000Z	[]	true	20	true	true	true	0		
2019-01-01T00:01:00.000Z	[]	true	20	true	true	true	0		
2019-01-01T00:02:00.000Z	[]	true	20	true	true	true	0.047619047619047616		
2019-01-01T00:03:00.000Z	[]	true	20	true	true	true	0.047619047619047616		
2019-01-01T00:04:00.000Z	[]	true	20	true	true	true	0.14284582683332223		
2019-01-01T00:05:00.000Z	[]	true	20	true	false	true	0.23805754488548037		
2019-01-01T00:06:00.000Z	[]	true	20	true	false	true	0.28565776603128423		
2019-01-01T00:07:00.000Z	[]	true	20	true	false	true	0.3332542339422178		
2019-01-01T00:08:00.000Z	[]	true	20	true	false	true	0.3808469526254273		
2019-01-01T00:09:00.000Z	[]	true	20	true	false	true	0.42843592608259695		
2019-01-01T00:10:00.000Z	[]	true	20	true	false	true	0.47602115830995684		
2019-01-01T00:11:00.000Z	[]	true	20	true	false	true	0.523602653290829		
2019-01-01T00:12:00.000Z	[]	true	20	true	false	true	0.5711804150329409		
2019-01-01T00:13:00.000Z	[]	true	20	true	false	true	0.6187544474938218		
2019-01-01T00:14:00.000Z	[]	true	20	true	false	true	0.666324754655421		
2019-01-01T00:15:00.000Z	[]	true	20	true	false	true	0.7138913404868101		
2019-01-01T00:16:00.000Z	[]	true	20	true	false	true	0.7614542089516521		
2019-01-01T00:17:00.000Z	[]	true	20	true	false	true	0.8090133640082078		
2019-01-01T00:18:00.000Z	[]	true	20	true	false	true	0.8565688096093442		
2019-01-01T00:19:00.000Z	[]	true	20	true	false	true	0.9041205497025 42		
2019-01-01T00:20:00.000Z	[]	true	20	true	false	true	0.9516685882299025		
2019-01-01T00:21:00.000Z	[]	true	20	true	false	true	0.9992129291281551		
2019-01-01T00:22:00.000Z	[]	true	20	true	false	true	1.0467535763286653		

Figure 5.1: Motion sensor and temperature sensor synchronization

5.1.2 Use Case 2: complex device behavior and activity recognition

The researcher wants to create a set of conditions-actions without manually configuring each device to fulfill the researcher's goal. He intends to model devices capable of recognizing movement within the smart environment, and if possible, ease the inhabitants living conditions within the smart home.

- Close blinds during the night time;

- Turn the light on when someone is nearby the respective device;
- Turn off the air conditioner if the fireplace is active.

The simulator must be able to accommodate the set of rules that the researcher wants to integrate within the virtual smart home, abstracting to him the underlying communication. The behaviors specified have different characteristics with the intent to test the rules and things behavior under different constraints.

- **Simulation details:** Section C.2 (p. 87) includes the partial representation of the scenario created for this use case. The focus of the scenario is on the rules and physical properties of the virtual system, so the description presents the configuration for the rules mentioned above and two physical properties: temperature and luminosity;
- **Expected behavior:** The rules should activate and deactivate according to the specified conditions, having an awareness of the simulation context (i.e., time, environmental properties, movement within the smart home);
- **Analysis:** The scenario shows that the devices' behavior is independent of the agents' interaction and can be associated with the context of the smart environment (*cf.* Figure 5.2), whereas the synchronization between the simulator components is seamless. The rules engine provides the capability to create complex device behavior, whereas the type of triggers and effects give flexibility to the user to create an extensive set of behaviors. The rule triggers and effects can be associated with the devices state or with the simulation context, which embodies an advantage when compared to the current tools that only consider the devices state. The use case addresses the desiderata related to the easy configuration of devices specified in Section 3.4.1 (p. 26).

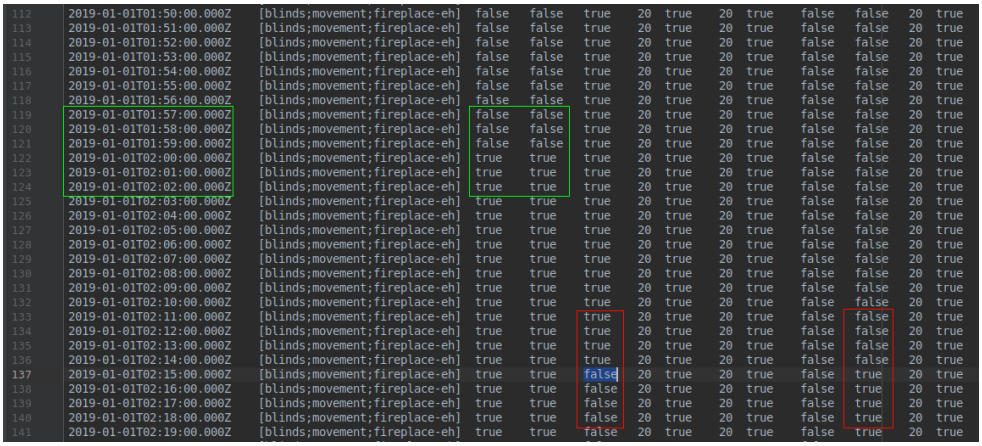


Figure 5.2: Device behavior within the virtual home

5.1.3 Use Case 3: environment with simulation of physical properties

The researcher wants to understand the influence of physical properties on the devices, so he aims to model physical properties such as temperature, light, and pressure to create a simulation context akin to a real-life scenario. For that, he needs to configure devices capable of measuring the physical properties of the simulation and configure the sensors to adapt to the physical environment:

- A rule ensures that the light sensor is off if the luminosity property is superior to a certain threshold;
- The virtual environment has devices that sense the current temperature and respond according to it;
- A rule in the system closes all the blinds if the external wind speed is higher than a specified threshold.

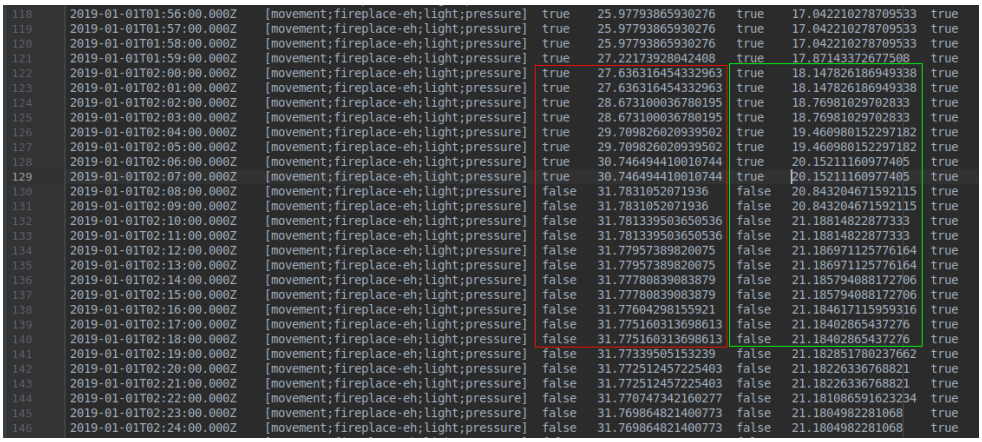
The researcher wants to understand how the physical properties spread in the virtual environment and if that flow is realistic, taking into account the context configured.

- **Simulation details:** The context is the same as the one created for the Use Case 2;
- **Expected behavior:** The devices must have the capability to sense and alter the physical properties of the environment. Each entity absorption and diffusion rates must affect the properties flow within the virtual simulation, whereas the rates depend on the current state of the entity (*cf.* Figure 5.3, p. 53);
- **Analysis:** The tool successfully implements the simulation of properties using a cellular automaton approach with a seamless interplay between the devices state and the conditions of the virtual world. Nevertheless, physical properties in a real-world context have substantially different equations that define their propagation and behave differently when in contact with the environmental surroundings (i.e., a blind stops the wind however heat propagates through it). Since the tool uses the same approach indiscriminately of the property, the modeling realism is severely impaired and simplified. The use case addresses the desiderata related to simulation of physical properties specified in Section 3.4.4 (p. 27).

5.1.4 Use Case 4: agent dynamic behavior

The researcher wants to add three persons to the simulation: himself, his wife, and his elderly father. The researcher must be able to configure the agents to represent their real-life behaviors easily:

- He has a straightforward daily routine, having eating, sleeping, and working as his main activities. He works with the surrounding lights turned on;
- The wife has the same activities as the husband but gives priority to different necessities. When the husband is working, she prefers to enjoy leisure time;



118	2019-01-01T01:56:00.000Z	[movement;fireplace-eh;light;pressure]	true	25.97793865930276	true	17.042210278709533	true
119	2019-01-01T01:57:00.000Z	[movement;fireplace-eh;light;pressure]	true	25.97793865930276	true	17.042210278709533	true
120	2019-01-01T01:58:00.000Z	[movement;fireplace-eh;light;pressure]	true	25.97793865930276	true	17.042210278709533	true
121	2019-01-01T01:59:00.000Z	[movement;fireplace-eh;light;pressure]	true	27.22173928042408	true	17.87143372677508	true
122	2019-01-01T02:00:00.000Z	[movement;fireplace-eh;light;pressure]	true	27.636316454332963	true	18.147826186949338	true
123	2019-01-01T02:01:00.000Z	[movement;fireplace-eh;light;pressure]	true	27.636316454332963	true	18.147826186949338	true
124	2019-01-01T02:02:00.000Z	[movement;fireplace-eh;light;pressure]	true	28.673100036780195	true	18.76981029702833	true
125	2019-01-01T02:03:00.000Z	[movement;fireplace-eh;light;pressure]	true	28.673100036780195	true	18.76981029702833	true
126	2019-01-01T02:04:00.000Z	[movement;fireplace-eh;light;pressure]	true	29.709826020939502	true	19.460980152297182	true
127	2019-01-01T02:05:00.000Z	[movement;fireplace-eh;light;pressure]	true	29.709826020939502	true	19.460980152297182	true
128	2019-01-01T02:06:00.000Z	[movement;fireplace-eh;light;pressure]	true	30.746494410010744	true	20.15211160977405	true
129	2019-01-01T02:07:00.000Z	[movement;fireplace-eh;light;pressure]	true	30.746494410010744	true	20.15211160977405	true
130	2019-01-01T02:08:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.7831052071936	false	20.843204671592115	true
131	2019-01-01T02:09:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.7831052071936	false	20.843204671592115	true
132	2019-01-01T02:10:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.781339503650536	false	21.18814822877333	true
133	2019-01-01T02:11:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.781339503650536	false	21.18814822877333	true
134	2019-01-01T02:12:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.77957389820075	false	21.186971125776164	true
135	2019-01-01T02:13:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.77957389820075	false	21.186971125776164	true
136	2019-01-01T02:14:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.77780839083879	false	21.185794088172706	true
137	2019-01-01T02:15:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.77780839083879	false	21.185794088172706	true
138	2019-01-01T02:16:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.77604298155921	false	21.184617115959316	true
139	2019-01-01T02:17:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.775160313698613	false	21.18402865437276	true
140	2019-01-01T02:18:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.775160313698613	false	21.18402865437276	true
141	2019-01-01T02:19:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.77339505153239	false	21.182851780237662	true
142	2019-01-01T02:20:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.772512457225403	false	21.18226336768821	true
143	2019-01-01T02:21:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.772512457225403	false	21.18226336768821	true
144	2019-01-01T02:22:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.770747342160277	false	21.181086591623234	true
145	2019-01-01T02:23:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.769864821400773	false	21.1804982281068	true
146	2019-01-01T02:24:00.000Z	[movement;fireplace-eh;light;pressure]	false	31.769864821400773	false	21.1804982281068	true

Figure 5.3: Outcome of a rule with multi triggers and multi effects related to the wind speed

- The father has a short set of activities compared to the above agents.

The researcher intends to analyze how the simulator portrays the defined behavior, the inherent realism, and how the simulator addresses the presence of multiple inhabitants.

- **Simulation details:** Section C.3 (p. 91) contains the partial representation of the context created for this scenario. There are three agents within the virtual home with a different set of activities and necessities. The virtual home has five divisions with several home appliances where the agents perform their activities: 2 bedrooms, a bathroom, a common area, and a kitchen (*cf.* Figure 5.4, p. 54);
- **Expected behavior:** The agents should perform their activities with the intent to fulfill their needs. In the case that an entity is unavailable, the agent should perform another activity;
- **Analysis:** As seen by the configuration of the agents, it is possible to describe an extensive set of behaviors. The use of complex logic, although increasing the complexity, offer more flexibility in terms of the activities and their interaction with the context of the simulation. The simulator does not support interactions between agents, which undermines the modeling of an essential part of human behavior; however, it is capable of handling multi-inhabitants since the agents perform activities only when the respective entities are available. The use case addresses the desiderata related to the extensible representation of human behavior, described in Section 3.4.5 (p. 27), and the modeling of the virtual home context as used in model-based approaches.

5.1.5 Use Case 5: interactive simulation and data generation

The researcher needs to control the simulation settings during run-time to analyze the effects of each change on the virtual home, especially the response to abnormal and erratic behavior.

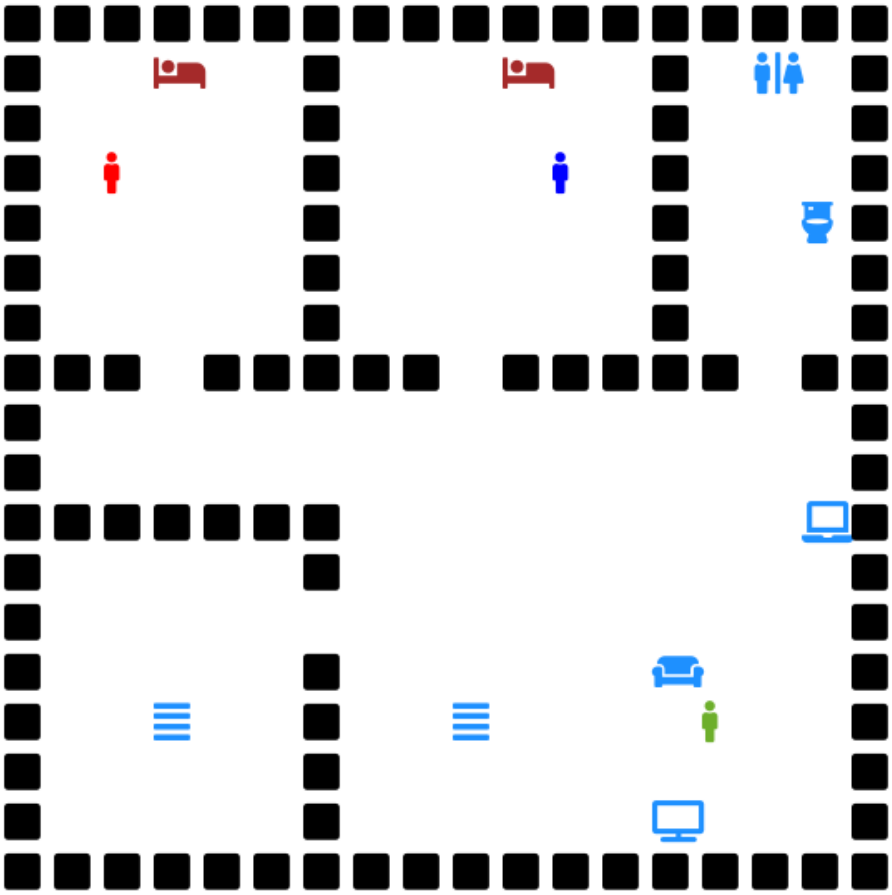


Figure 5.4: Simulation context with multiple inhabitants

The researcher wants to see the actions taken by the modeled agents and how they influence the environment. It is also important for him to see how environmental properties change as time goes on:

- Select agents' next activity;
- Change the things internal properties;
- Activate or deactivate rules;
- Update the environment properties, such as temperature, and check how the system reacts to such changes.

On the other hand, he intends to do post analysis of the simulation data, to check how the specified rules influence the overall system and the effect of each activity within the environment.

- **Simulation details:** The context is the one represented in Section C.1 (p. 81);

- **Expected behavior:** The agent should perform his current activity and then start the activity defined by the user; the properties of the thing should synchronize automatically with the entities and influence the simulation context; the update of the source properties affect the nearby entities immediately;
- **Analysis:** The simulator provides a hybrid approach, with modeling and interactivity. The latter is, however, limited compared to fully interactive tools, as it only allows to alter a few aspects of the context during run time (*cf.* Figure 5.5). Nevertheless, the synchronization of the context is swift, and the effects in the context are easily identifiable. The physical properties overlay and the tooltip helper of each entity ease the understanding of the simulation context. The use case addresses the desiderata related to the integration of a hybrid approach, incorporating the advantages of model-based and interactive approaches, as described in Section 3.4.3 (p. 27).

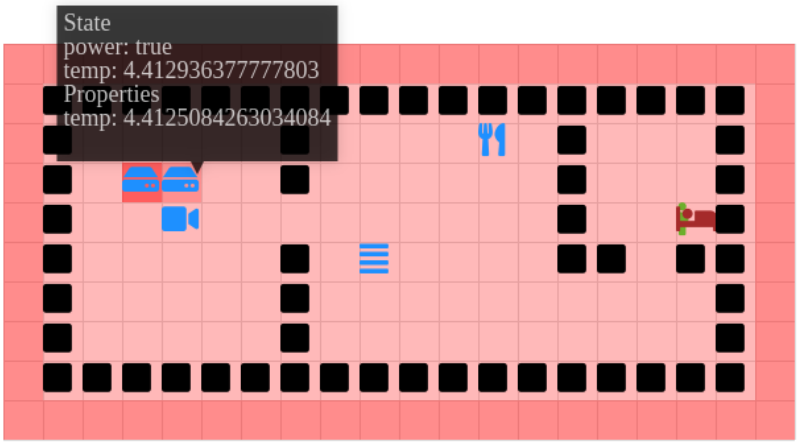


Figure 5.5: Overlay and tooltip helper

5.1.6 Use Case 6: dynamic layout

It is essential that the researcher can replicate the physical space of his smart house within the simulation as similar to real-life as possible, independently of the inherent complexity of the context and layout.

- **Simulation details:** The scenarios represented in Appendix C (p. 81) represent the broad range of contexts potentially specified in the simulator;
- **Analysis:** The 2D display and representation simplify the layout creation and underlying representation. However, such abstraction thwarts the simulator's capability to represent all

the intricacies of a smart home. In terms of usability, the user can quickly design a virtual smart home with an acceptable level of abstraction, whereas such a process is time-consuming with a layout more complex. The use case addresses the desiderata related to the flexible customization of a virtual scenario as specified in Section 3.4.2 (p. 26).

5.1.7 Use Case 7: simulation time configuration

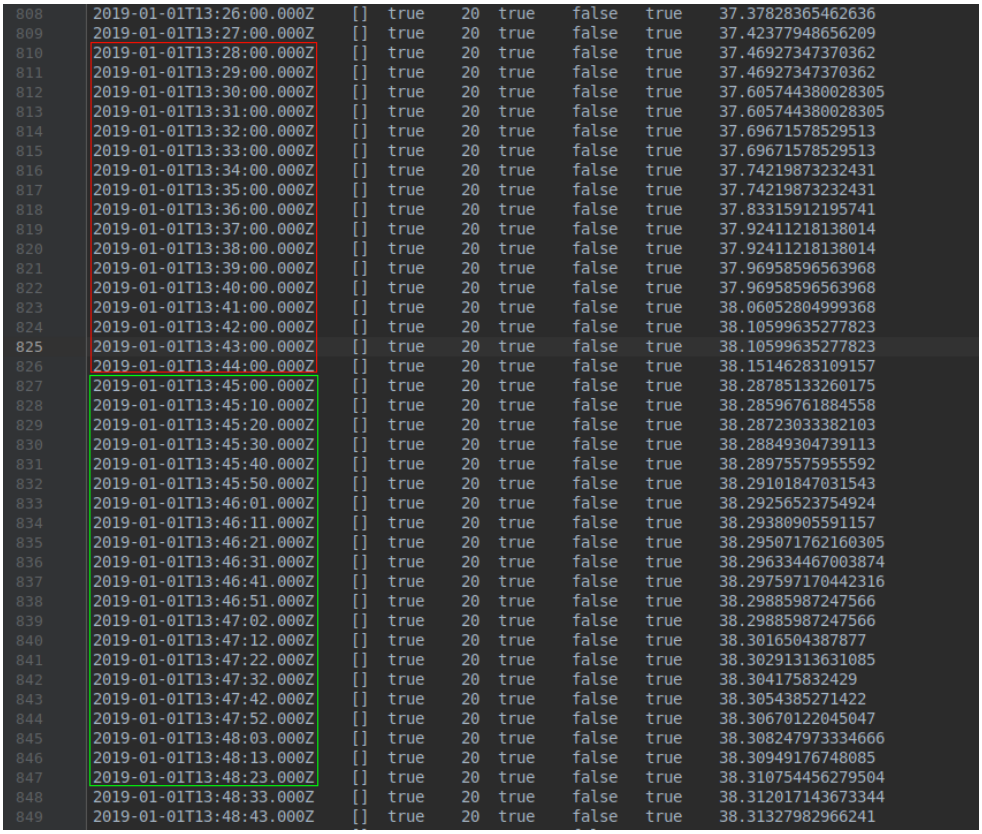
The researcher desires to simulate with standard speed settings during the day when the amount of activity within the home is higher and increase the speed at night when the lack of activity is apparent.

- **Simulation details:** The simulation processes one full day. During the daytime, the simulation has an update rate of 1 and an internal simulation increment of 10 seconds. During the night time, the simulation has an update rate of 45 and an internal simulation increment of 60 seconds;
- **Analysis:** It is possible to configure the update rate of the simulation, which modifies substantially the time needed to observe the simulation. Regarding the simulation internal time increment, the time granularity can be updated during run time, which allows a higher control over the detail obtained in the datasets (*cf.* Figure 5.6, p. 57). The range defined for the update rate and time increment is reasonable. The visualization quality of the simulation, independently of the time configuration, is not affected. The use case addresses the desiderata related to the integration of a hybrid approach specified in Section 3.4.3 (p. 27), especially the capability to control the internal simulation time setting just like interactive approaches.

5.2 Conclusions

As mentioned at the beginning of the chapter, the evaluation process and analysis was undertaken by the authors, who, throughout the process, tried to evaluate the tool from an objective standpoint.

The simulated scenarios show the simulator's capabilities and the full range of customization provided to the users. Compared to other tools which usually target a specific domain, the developed tool is sufficiently generic and flexible. However, its flexibility, as seen by the scenarios created, increases the abstraction level of the features and the level of detail of the virtual context. Section 6.4 (p. 61) details how future improvements could address the problems encountered in the evaluation process.



808	2019-01-01T13:26:00.000Z	[]	true	20	true	false	true	37.37828365462636
809	2019-01-01T13:27:00.000Z	[]	true	20	true	false	true	37.42377948656209
810	2019-01-01T13:28:00.000Z	[]	true	20	true	false	true	37.46927347370362
811	2019-01-01T13:29:00.000Z	[]	true	20	true	false	true	37.46927347370362
812	2019-01-01T13:30:00.000Z	[]	true	20	true	false	true	37.605744380028305
813	2019-01-01T13:31:00.000Z	[]	true	20	true	false	true	37.605744380028305
814	2019-01-01T13:32:00.000Z	[]	true	20	true	false	true	37.69671578529513
815	2019-01-01T13:33:00.000Z	[]	true	20	true	false	true	37.69671578529513
816	2019-01-01T13:34:00.000Z	[]	true	20	true	false	true	37.74219873232431
817	2019-01-01T13:35:00.000Z	[]	true	20	true	false	true	37.74219873232431
818	2019-01-01T13:36:00.000Z	[]	true	20	true	false	true	37.83315912195741
819	2019-01-01T13:37:00.000Z	[]	true	20	true	false	true	37.92411218138014
820	2019-01-01T13:38:00.000Z	[]	true	20	true	false	true	37.92411218138014
821	2019-01-01T13:39:00.000Z	[]	true	20	true	false	true	37.96958596563968
822	2019-01-01T13:40:00.000Z	[]	true	20	true	false	true	37.96958596563968
823	2019-01-01T13:41:00.000Z	[]	true	20	true	false	true	38.06052804999368
824	2019-01-01T13:42:00.000Z	[]	true	20	true	false	true	38.10599635277823
825	2019-01-01T13:43:00.000Z	[]	true	20	true	false	true	38.10599635277823
826	2019-01-01T13:44:00.000Z	[]	true	20	true	false	true	38.15146283109157
827	2019-01-01T13:45:00.000Z	[]	true	20	true	false	true	38.28785133260175
828	2019-01-01T13:45:10.000Z	[]	true	20	true	false	true	38.28596761884558
829	2019-01-01T13:45:20.000Z	[]	true	20	true	false	true	38.28723033382103
830	2019-01-01T13:45:30.000Z	[]	true	20	true	false	true	38.28849304739113
831	2019-01-01T13:45:40.000Z	[]	true	20	true	false	true	38.28975575955592
832	2019-01-01T13:45:50.000Z	[]	true	20	true	false	true	38.29101847031543
833	2019-01-01T13:46:01.000Z	[]	true	20	true	false	true	38.29256523754924
834	2019-01-01T13:46:11.000Z	[]	true	20	true	false	true	38.29380905591157
835	2019-01-01T13:46:21.000Z	[]	true	20	true	false	true	38.295071762160305
836	2019-01-01T13:46:31.000Z	[]	true	20	true	false	true	38.296334467003874
837	2019-01-01T13:46:41.000Z	[]	true	20	true	false	true	38.297597170442316
838	2019-01-01T13:46:51.000Z	[]	true	20	true	false	true	38.29885987247566
839	2019-01-01T13:47:02.000Z	[]	true	20	true	false	true	38.29885987247566
840	2019-01-01T13:47:12.000Z	[]	true	20	true	false	true	38.3016504387877
841	2019-01-01T13:47:22.000Z	[]	true	20	true	false	true	38.30291313631085
842	2019-01-01T13:47:32.000Z	[]	true	20	true	false	true	38.304175832429
843	2019-01-01T13:47:42.000Z	[]	true	20	true	false	true	38.3054385271422
844	2019-01-01T13:47:52.000Z	[]	true	20	true	false	true	38.30670122045047
845	2019-01-01T13:48:03.000Z	[]	true	20	true	false	true	38.308247973334666
846	2019-01-01T13:48:13.000Z	[]	true	20	true	false	true	38.30949176748085
847	2019-01-01T13:48:23.000Z	[]	true	20	true	false	true	38.310754456279504
848	2019-01-01T13:48:33.000Z	[]	true	20	true	false	true	38.312017143673344
849	2019-01-01T13:48:43.000Z	[]	true	20	true	false	true	38.31327982966241

Figure 5.6: Time granularity

Chapter 6

Conclusions

This chapter summarizes the work developed throughout this project. Section 6.1 explains the main difficulties encountered during the development and how they were surpassed, while Section 6.2 (p. 60) overviews the contributions to the IoT simulation ecosystem. Section 6.3 (p. 61) summarizes the project and the conclusions gathered. Finally, Section 6.4 (p. 61) outlines the future work that could be implemented, which, in the authors' opinion, would considerably improve the usability and usefulness of the tool.

6.1 Challenges

Throughout the report, the challenges outlined in Section 3.5 (p. 27) were addressed. A challenge does not have a unique approach considered fail-proof, but a multitude of ways to address the problem with many advantages and disadvantages. The authors usually adopted a middle-ground approach intending to balance the advantages and disadvantages of the final solution, taking into account the possible future requirements and the time constraint of the project.

The level of usability and flexibility of the system appeared as the primary challenge. The flexibility-usability tradeoff is a design principle specifying that as the flexibility of a system increases, its usability decreases. Usually, more flexibility requires a broader set of requirements, which results in complexity and usability compromises. Usually, the higher the flexibility and level of abstraction, the model may become too unnatural and lose the intricacies of a realistic context. On the other hand, flexibility directly impacts the usability of the simulator, which can steer the users away from the tool. To accommodate all the configuration aspects, the authors selected the flexibility of the solution as the more significant force constraining the solution and opted for the use of JSON to allow the design of the smart home and its context. Effectively, such an approach is not ideal for a quick configuration nor gives the user an intuitive interface for this task as a well-designed user interface would give.

The IoT environment and the vast diversity of devices available is a challenge faced in the research area of simulators applied to IoT. This diversity entails the need to configure the simulator to support a wide range of devices and their interactions within the environment. Despite Mozilla's attempt at a Web Thing API, the lack of a standard interface for IoT devices means that the flexibility given to the users in terms of devices customization is directly dependent and constrained by the API integrated. Thus, the same limitation applies to this work.

A smart home has an inherently complex context. The design of a smart home in a simulator can, in theory, integrate its unique details. However, the process to do so is cumbersome for the users. For example, the design of a smart home in 3D allows fine-grained control over the design and effectively the scenario is akin to a real-life one. This approach has several disadvantages: the design becomes cumbersome for the user and the underlying model to represent the design and its details scales to a new complexity. Faced with this challenge, the authors opted for a simple underlying model with the capability to tackle several requirements and design with a higher level of abstraction.

The physical environment conditions are hard to simulate realistically, as each physical property is defined by complex equations, e.g., the heat transference uses thermodynamics equations. There are engines with the capability to model such physical phenomena; however, their underlying models have high computational demands. Thus, the solution used tries to be flexible and incorporate an approach that simulates the physical conditions more straightforwardly via a CA.

6.2 Contributions

As mentioned in Chapter 3 (p. 23), the primary focus of this dissertation is the creation of a flexible simulator capable of creating and testing a broad range of real scenarios incorporating several features that hitherto no simulator appeared to incorporate simultaneously. As such, the main contributions of the dissertation are the following:

- **Integration of a physics engine:** The simulator integrates a simplistic physics engine, opening the possibility to configure the physical properties of the environment within a virtual smart home;
- **Seamless interplay between devices and surrounding dynamic context:** The simulator presents an architecture where the main components are decoupled between themselves; Nevertheless, it is capable of integrating complex device behavior related to the intrinsic context of the simulation environment or the state of other devices without visible hindrance to the visualization;
- **Interactive visualization and data generation:** The simulator integrates the main advantages of model-based and interactive approaches. Although presenting a 2D viewer, the user can manage the simulation during run-time, interact with the context, and control the data generation granularity.

6.3 Conclusions

The framework overview shows that it offers a comprehensive set of features, allowing users to design and simulate smart home scenarios with distinct characteristics without yielding excessively in terms of detail granularity. The hybrid approach integrated leverages the advantages of both model-based and interactive approaches enabling the user to have an active role during the simulation modeling and run time alike. On the other hand, the capability to represent complex logic related to the dynamic context of the simulation gives flexibility to the user to recreate scenarios with intricate details.

The project attains the requirements introduced in Section 3.4 (p. 25), which can be inferred by the analysis of the evaluation made in Chapter 5 (p. 49). The authors deem the features implemented not as a final solution but rather a starting point for the possibility of approaches applied to a domain so intricate as the simulation of smart homes. The project developed, because of its contributions and discussion revolving the modeling and experimentation of virtual environments applied to smart homes, represents a valuable addition to the plethora of simulation tools currently available in the state-of-the-art.

As mentioned before, the flexibility of the solution is the primary force of the project. Throughout the development, the authors realized that a solution must have the right balance between flexibility and usability. The authors consider that the developed tool has an acceptable level of flexibility and is easily extended with further features; however, the usability and overall user experience is flawed and could be enhanced.

The implementation of this simulator shows that the creation of a simulation engine for smart home environments must be an iterative process. The overall complexity of smart homes and their inhabitants is exceptionally high, which hinders the development of these tools. However, by creating an open-source tool with an architecture designed to incorporate future requirements, opens the possibility to iterate over the tool and enhance it. The developers and researchers must strive to create their tools for the open source community, as it creates a thriving environment around the solutions.

6.4 Future Work

The nature of the domain dealt with by the simulator opens the way to several improvements since every component of the solution will never be perfect but can always be iterated and enhanced. In the authors' opinion, the solution developed is a stepping stone for the possibility of extensions and features susceptible to integrate in the future:

- **Support of physical and simulated devices:** The architecture was built with the intent to support physical and simulated devices; however, the solution does not yet fulfill this requirement. Overall, the capability to support both types would increase the usefulness of the simulator, as it could be used to test and validate smart devices and their configuration in a real smart home.

- **Physical properties engine:** The simulation of physical properties has room to evolve, as there are in the literature examples of CA models with the representation of complex physical processes. Even if the computational requirements would increase, the potential to parallelize the automata could diminish the computational requirements of the physics engine. Besides, different properties require different modeling processes, so it would be a valuable addition to have an underlying simulation model for each of the principal physical properties that influence the context of a smart home (e.g., luminosity, temperature, pressure).

- **Usability:** The user interface is one of the main shortcomings of the application because of the tradeoff between usability and flexibility — a better user experience would substantially improve the ease of configuring the simulation. On the other hand, the virtual smart home layout is exceptionally simplistic. The integration of a virtual smart home UI with a higher level of detail would increase the usability of the simulator, however, in the authors' opinion, to maintain an underlying model simple to use and develop, the layout should remain a 2D environment.

- **Multi-agent paradigm:** The approach used for the behavior of agents depends entirely on the necessities defined, and the engine controls their processing throughout the simulation. In contrast, with a multi-agent paradigm with parallel computing, the agents could act as independent units deployed on cloud-computing instances, communicating with the simulation engine via a familiar interface. This approach would imbue the framework with the concept of cloud computing and allow the simulation to be deployable to the cloud and highly scalable.

- **Iterative machine learning techniques:** The simulator, alongside the simulation processing, writes the context to a dataset for posterior analysis. It would be interesting to integrate within the simulator machine learning techniques and algorithms to iterate over the data generated in so far and provide insights for the user about the overall design and efficiency of the system, such as information related to the configuration of devices and the context influence on the system rules.

References

[AAS+17] Nasser Alshammari, Talal Alshammari, Mohamed Sedky, Justin Champion, and Carolin Bauer. Openshs: Open smart home simulator. *Sensors*, 17:1003, 05 2017.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[AR07] I. Armac and D. Retkowitz. Simulation of smart environments. In *IEEE International Conference on Pervasive Services*, pages 257–266, July 2007.

[AR16] Muhammad Asadullah and Ahsan Raza. An overview of home automation systems. In *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, 12 2016.

[ARA12] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali. A review of smart homes—past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1190–1203, Nov 2012.

[ARCL13] A. Ariani, S. J. Redmond, D. Chang, and N. H. Lovell. Simulation of a smart home environment. In *2013 3rd International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering (ICICI-BME)*, pages 27–32, Nov 2013.

[BABB10] Kevin Bouchard, Amir Ajroud, Bruno Bouchard, and Abdenour Bouzouane. Simact: A 3d open source smart home simulator for activity recognition. In *AST/UCMA/ISA/ACN*, 2010.

[BDPP15] Alessio Botta, Walter Donato, Valerio Persico, and Antonio Pescapè. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, 56, 10 2015.

[BHB10] James Byrne, Cathal Heavey, and P.J. Byrne. A review of web-based simulation and supporting tools. *Simulation Modelling Practice and Theory*, 18(3):253 – 276, 2010.

[BJC09] J. Bruneau, W. Jouve, and C. Consel. Diasim: A parameterized simulator for pervasive computing applications. In *2009 6th Annual International Mobile and Ubiquitous Systems: Networking Services, MobiQuitous*, pages 1–10, July 2009.

[BKK11] Mario Buchmayr, Werner Kurschl, and Josef Küng. A simulator for generating and visualizing sensor data for ambient intelligence environments. *Procedia Computer Science*, 5:90 – 97, 2011. The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011).

[BMZA12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[CC03] A. K. R. Chowdhury and R. Chellappa. A factorization approach for activity recognition. In *2003 Conference on Computer Vision and Pattern Recognition Workshop*, volume 4, pages 41–41, June 2003.

[CCTK13] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, July 2013.

[CP07] J. Counsell and M. Puybaraud. Real-time data and analysis of the use of office space. In *2007 11th International Conference Information Visualization (IV '07)*, pages 579–583, July 2007.

[dBAMI05] Ruyter de BER, EHL Aarts, Panos Markopoulos, and Wijnand Ijsselsteijn. Ambient intelligence research in homelab: Engineering the user experience. *Electronics Letters - ELECTRON LETT*, 01 2005.

[DCPF18] J. P. Dias, F. Couto, A. C. R. Paiva, and H. S. Ferreira. A brief overview of existing tools for testing the internet-of-things. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 104–109, April 2018.

[DFF18] J. P. Dias, J. P. Faria, and H. S. Ferreira. A reactive and model-based approach for developing internet-of-things systems. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 276–281, Sep. 2018.

[DHSR08] George Demiris, Brian K. Hensel, Marjorie Skubic, and Marilyn Rantz. Senior residents' perceived need of and preferences for "smart home" sensor technologies. *International Journal of Technology Assessment in Health Care*, 24(1):120–124, 2008.

[DNMHD09] C D. Nugent, Maurice Mulvenna, Xin Hong, and Steven Devlin. Experiences in the development of a smart lab. *International Journal of Biomedical Engineering and Technology*, 2, 08 2009.

[EMMN18] Macarena Espinilla, Luis Martinez, J Medina, and C Nugent. The experience of developing the ujami smart lab. *IEEE Access*, PP:1–1, 06 2018.

[ERA09] Markus Eisenhauer, Peter Rosengren, and Pablo Antolin. A development platform for integrating wireless devices and sensors into ambient intelligence systems. In *2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, volume 2010, pages 1 – 3, 07 2009.

[GMBT04] B Georis, M Maziere, F Bremond, and M Thonnat. A video interpretation platform applied to bank agency monitoring. In *in Intelligent Distributed Surveillance Systems Workshop*, pages 46 – 50, 03 2004.

[Hew77] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323 – 364, 1977.

[IWSK07] Y. Ivanov, C. Wren, A. Sorokin, and I. Kaur. Visualizing the history of living spaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1153–1160, Nov 2007.

[JRM11] Zahra Forootan Jahromi, Amir Rajabzadeh, and Ali Reza Manashty. A multi-purpose scenario-based simulator for smart house environments. *CoRR*, abs/1105.2902, 2011.

[KH16] Rohan Kar and Rishin Haldar. Applying chatbots to the internet of things: Opportunities and architectural elements. *International Journal of Advanced Computer Science and Applications*, 7(11), 2016.

[KP13] B. Kormányos and B. Pataki. Multilevel simulation of daily activities: Why and how? In *2013 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 1–6, July 2013.

[LCK08] Jumphon Lertlakkhanakul, Jin Won Choi, and Mi Yun Kim. Building data model and simulation platform for spatial interaction management in smart home. *Automation in Construction*, 17(8):948 – 957, 2008.

[LCL+15] J. W. Lee, S. Cho, S. Liu, K. Cho, and S. Helal. Persim 3d: Context-driven simulation and modeling of human activities in smart spaces. *IEEE Transactions on Automation Science and Engineering*, 12(4):1243–1256, Oct 2015.

[LSJN15] J. Lundström, J. Synnott, E. Järpe, and C. D. Nugent. Smart home simulation using avatar control and probabilistic sampling. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 336–341, March 2015.

[MG11] Peter Mell and Timothy Grance. The nist definition of cloud computing. Technical Report 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD, September 2011.

[MOG+10] Dr. Kris Mcglinn, Eleanor Oneill, Alan Gibney, Declan O'Sullivan, and David Lewis. Simcon: A tool to support rapid evaluation of smart building application design using context simulation and virtual reality. *J. UCS*, 16:1992–2018, 01 2010.

[MSPC12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10:1497–1516, 2012.

[NJD09] T. V. Nguyen, Jin Gook Kim, and Deokjai Choi. Iss: The interactive smart home simulator. In *2009 11th International Conference on Advanced Communication Technology*, volume 03, pages 1828–1833, Feb 2009.

[NPF+11] Norbert Noury, Julien Poujaud, Anthony Fleury, Ronald Nocua, Tareq Haddidi, and Pierre Rumeau. Smart sweet home... a pervasive environment for sensing our daily activity? *Activity Recognition in Pervasive Intelligent Environments*, 4, 01 2011.

REFERENCES

[NYT+06] Hiroshi Nishikawa, Shinya Yamamoto, Morihiko Tamai, Kouji Nishigaki, Tomoya Kitani, Naoki Shibata, Keiichi Yasumoto, and Minoru Ito. Ubireal: Realistic smartspace simulator for systematic testing. In Paul Dourish and Adrian Friday, editors, *UbiComp 2006: Ubiquitous Computing*, pages 459–476, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[PMHY07] J. Park, M. Moon, S. Hwang, and K. Yeom. Cass: A context-aware simulation system for smart home. In *5th ACIS International Conference on Software Engineering Research, Management Applications (SERA 2007)*, pages 461–467, Aug 2007.

[QPC+11] Qiang Fu, Ping Li, Cai Chen, Lanlan Qi, Yongqiang Lu, and Chen Yu. A configurable context-aware simulator for smart home systems. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 39–44, Oct 2011.

[ROL+13] R. A. Ramlee, M. A. Othman, M. H. Leong, M. M. Ismail, and S. S. S. Ranjit. Smart home system using android application. In *2013 International Conference of Information and Communication Technology (ICoICT)*, pages 277–280, March 2013.

[SAH+13] Moataz Soliman, Tobi Abiodun, Tarek Hamouda, Jiehan Zhou, and Chung-Horng Lung. Smart home: Integrating internet of things with web services and cloud computing. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 2, pages 317–320, 12 2013.

[SCNM14] J. Synnott, L. Chen, C. D. Nugent, and G. Moore. The creation of simulated activity datasets using a graphical intelligent environment simulation tool. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4143–4146, Aug 2014.

[SDPA14] Priya Suresh, Jérémie Daniel, Velusamy Parthasarathy, and R. H. Aswathy. A state of the art review on the internet of things (iot) history, technology and fields of deployment. *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, pages 1–8, 2014.

[vdM17] Rob van der Meulen. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016, February 7, 2017. [Online; accessed 01-February-2019].

[Vic84] Gérard Y. Vichniac. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):96 – 116, 1984.

[VPCG16] A. Vasilateanu, I. A. Popescu, A. S. Cergan, and N. Goga. Smart home simulation system. In *2016 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–5, Oct 2016.

[XYWV12] Feng Xia, Laurence T. Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101–1102, 2012.

[Yam07] Tatsuya Yamazaki. The ubiquitous home. *International Journal of Smart Home*, 1, 02 2007.

[ZCW+14] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen, and S. Shieh. Iot security: Ongoing challenges and research opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 230–234, Nov 2014.

[ZKHS17] M. Zouai, O. Kazar, B. Haba, and H. Saouli. Smart house simulation based multi-agent system and internet of things. In *2017 International Conference on Mathematics and Information Technology (ICMIT)*, pages 201–203, Dec 2017.

REFERENCES

Appendix A

Simulation JSON Representations

This appendix includes samples of the main objects used in the framework developed.

A.1 Things

```
{
  id: "props-thing",
  name: "props-thing",
  description: "Thing with several properties",
  properties: {
    readOnlyProp: {
      type: "boolean",
      readOnly: true,
      value: true
    },
    minMaxProp: {
      type: "number",
      minimum: 10,
      maximum: 20,
      value: 15
    },
    enumProp: {
      type: "string",
      enum: ["val1", "val2", "val3"],
      value: "val2"
    },
    multipleProp: {
      type: "integer",
      minimum: 0,
```

```
      maximum: 600,
      value: 10,
      multipleOf: 5
    }
  }
};
```

Listing A.1: Thing with several properties

```
{
  id: "OnOffSwitch",
  name: "OnOffSwitch",
  description: "Thing with actions and events",
  "@type": ["OnOffSwitch"],
  properties: {
    power: {
      "@type": "OnOffProperty",
      type: "boolean",
      value: true,
    }
  },
  actions: {
    reboot: {
      description: "Reboot the device",
    }
  },
  events: {
    reboot: {
      description: "Going down for reboot",
    }
  }
};
```

Listing A.2: Thing with actions and events

A.2 Rules

```
{
  property: {
    type: "boolean",
    thingId: "light1",
```

```
    id: "on"
  },
  type: "PulseEffect",
  value: true
};
```

Listing A.3: Pulse effect

```
{
  property: {
    type: "number",
    thingId: "thermostat",
    id: "temp",
    unit: "celsius",
    description: "thermostat setpoint"
  },
  type: "SetEffect",
  value: 30
};
```

Listing A.4: Set effect

```
{
  property: {
    type: "number",
    thingId: "light2",
    id: "hue"
  },
  type: "LevelTrigger",
  levelType: "LESS",
  value: 120
};
```

Listing A.5: Level trigger

```
{
  property: {
    type: "boolean",
    thingId: "light1",
    id: "on"
  },
  type: "BooleanTrigger",
```

```
  onValue: true
};
```

Listing A.6: Boolean trigger

```
{
  property: {
    type: "string",
    thingId: "light2",
    id: "color"
  },
  type: "EqualityTrigger",
  value: "#ff7700"
};
```

Listing A.7: Equality trigger

A.3 Agents

```
[{
  name: "comfort",
  priority: 4,
  decay: {
    type: "RANGE",
    value: {
      minimum: 5,
      maximum: 10,
    },
    unit: "M",
  }
},
{
  name: "hunger",
  priority: 5,
  decay: {
    type: "RULE",
    value: [{
      conditions: {
        all: [{
          fact: "hour",
          operator: "greaterThan",
```

```
          value: 12
        }]
      },
      value: 1
    }, {
      conditions: {
        all: [{
          fact: "hour",
          operator: "lessThanInclusive",
          value: 12
        }]
      },
      value: 0.5
    }],
    unit: "H",
  }
},
{
  name: "hygiene",
  priority: 5,
  decay: {
    type: "NUMBER",
    value: 1,
    unit: "H",
  }
}]
```

Listing A.8: Necessity JSON representation

```
{
  name: "renato",
  position: {x: 5, y: 5},
  icon: "f245",
  color: "#6daf2b",
  activities: ["sleep", "eat", "work"],
  necessities: ["comfort", "hunger", "hygiene"],
};
```

Listing A.9: Agent JSON representation

A.4 Activities

```
{
  name: "getFood",
  duration: {
    value: 1,
    unit: "M",
  },
  entityGroup: "Fridges",
  effects: [],
  preconditions: [{ property: "open", value: true, type: "BOOLEAN" }
   ],
  postconditions: [{ property: "open", value: false, type: "BOOLEAN"
    }],
  dependencies: []
};
```

Listing A.10: Activity JSON representation

A.5 Physical Properties

```
{
  id: "temp",
  title: "Temperature",
  description: "Temperature property",
  unit: "celsius",
  limits: {
    minimum: 0,
    maximum: 30
  },
};
```

Listing A.11: Temperature metadata

```
{
  value: 5,
  absorptionRate: {
    type: "NUMBER",
    value: 0.25,
    unit: "M"
  },
```

```
  diffusionRate: {
    type: "NUMBER",
    value: 1,
    unit: "M"
  }
};
```

Listing A.12: Temperature diffusion and absorption rate

```
{
  id: "temp",
  title: "Temperature",
  description: "Temperature property",
  unit: "celsius",
  value: 5,
  limits: {
    minimum: 0,
    maximum: 30
  },
  absorptionRate: {
    type: "RULE",
    value: [{
      conditions: {
        all: [{
          fact: "open",
          operator: "equal",
          value: true
        }]
      },
      value: 1
    }, {
      conditions: {
        all: [{
          fact: "open",
          operator: "equal",
          value: false
        }]
      },
      value: 0.5
    }],
    unit: "M"
```

```
  },
  diffusionRate: {
    type: "NUMBER",
    value: 1,
    unit: "M"
  }
};
```

Listing A.13: Temperature representation with complex logic associated with a door

A.6 Entities

```
{
  properties: [{
    id: "temp",
    value: 5,
    absorptionRate: {
        type: "NUMBER",
        value: 0.25,
        unit: "M"
    },
    diffusionRate: {
        type: "NUMBER",
        value: 1,
        unit: "M"
    }
  }],
  state: [],
  color: "#FFFFFF",
  position: {
    x: 1,
    y: 1,
  },
  type: ["Static", "Walls"],
};
```

Listing A.14: Static entity

```
{
  properties: [],
  state: [{
```

```
    id: "light",
    title: "lightOnOff",
    description: "Light bulb",
    type: "string",
    value: "on",
    enum: ["on", "off"],
  }],
  color: "#FFFFFF",
  name: "Wall",
  position: {
    x: 1,
    y: 2,
  },
  type: ["Interactive"],
};
```

Listing A.15: Interactive entity representation

```
{
  properties: [],
  state: [{
    id: "light",
    title: "lightOnOff",
    description: "Light bulb",
    type: "string",
    value: "on",
    enum: ["on", "off"],
  }],
  thing: "validation-1",
  color: "#FFFFFF",
  name: "Wall",
  position: {
    x: 1,
    y: 2,
  },
  type: ["Interactive"],
};
```

Listing A.16: Interactive entity associated with a device

```
{
  "name": "wall",
```

```
  "type": ["Walls", "Static"],
  "icon": "f0c8",
  "color": "#000000",
  "walkable": false,
  "properties": [{
    "id": "temp",
    "absorptionRate": {
      "type": "NUMBER",
      "value": 1,
      "unit": "H"
    },
    "diffusionRate": {
      "type": "NUMBER",
      "value": 0.2,
      "unit": "H"
    }
  }],
  "position": [
    {"x": 14, "y": 2},
    {"x": 14, "y": 3},
    {"x": 14, "y": 4},
    {...},
  ]
}
```

Listing A.17: Multiple entities with same representation

Appendix B

Simulation Update Process

This appendix includes the function used to process a simulation step.

```
1  private async update() {
2    // If simulation is not running, remove any future timeout
3    if (this._state !== SimulationState.RUNNING) {
4      clearTimeout(this._time.timer);
5      return;
6    }
7
8    // If there are clients listening and the number of updates is equal to the value
       defined by the client, send the simulation state
9    if (this._wss && this._time.updateRate === this._time.updateCounter) {
10     this._wss.clients.forEach((client, ws) => {
11       if (ws.OPEN) {
12         client.send(JSON.stringify({type: "SimulationUpdate", data: this.
       toDescription()}));
13       }
14     });
15     this._time.updateCounter = 0;
16   }
17
18   // Write update to dataset
19   this.writer.writeData(this._time.simulationTime.toDate(), this._registry.
       getThings(), this._agents,
20     this._rulesEngine.getRules());
21
22   // Update internal simulation time
23   this._time.startUpdate();
24
25   // Get physical property data, for current time
26   const edgeData = this.getCustomData(JSON.stringify({
27     year: this._time.simulationTime.year,
28     month: this._time.simulationTime.month,
29     day: this._time.simulationTime.day,
30     hour: this._time.simulationTime.hour
```

```
31    }));
32
33    // Set the physical properties of all edge entities
34    const entities: InteractiveEntity[] = this.grid.interactiveCells.map(
        interactiveCellPos => {
35      return ((this.grid.getCell(interactiveCellPos) as Cell).entity) as
        InteractiveEntity;
36    });
37
38    // Synchronize properties
39    const auxThings: string[] = [];
40    for (const entity of entities) {
41      if (entity.thingId && this.thingsData.has(entity.thingId)) {
42        entity.setStateValues(this.thingsData.get(entity.thingId), false);
43        if (auxThings.findIndex(t => t === entity.thingId) === -1) {
44          auxThings.push(entity.thingId);
45        }
46      }
47    }
48    for (const t of auxThings) {
49      this.thingsData.set(t, new TSMap());
50    }
51
52    // Update agent
53    for (const agent of this.agents.values()) {
54      await agent.update(this);
55    }
56
57    // Update entities
58    const positions = this.agents.map(a => a.position);
59    await this._grid.update(edgeData, this._time, positions);
60
61    // Update rules that depend on the simulation context
62    await this.rulesEngine.update(this._time);
63
64    // Ends update and sets next update timeout
65    this._time.endUpdate(this.update.bind(this));
66  }
```

Listing B.1: Simulator update function

Appendix C

Simulation Scenarios JSON Representation

This appendix includes the JSON representation of the scenarios described in Chapter 5.

C.1 Scenario 1

```
{
  "things": [
    {
      "id": "electric-heater",
      "name": "Electric heater",
      "description": "Warms up a room",
      "@type": [
        "OnOffSwitch",
        "Emitter"
      ],
      "properties": {
        "power": {
          "@type": "OnOffProperty",
          "type": "boolean",
          "value": true
        },
        "temp": {
          "@type": "TemperatureProperty",
          "type": "number",
          "value": 20
        }
      }
```

```
    },
    {
      "id": "motion-sensor",
      "name": "Motion Sensor",
      "description": "Senses movement within a smart home",
      "@type": [
        "OnOffSwitch",
        "MotionSensor",
        "Sensor"
      ],
      "properties": {
        "power": {
          "@type": "OnOffProperty",
          "type": "boolean",
          "value": true
        },
        "motion": {
          "@type": "MotionProperty",
          "type": "boolean",
          "value": true
        }
      }
    },
    {
      "id": "temperature-sensor",
      "name": "Temperature Sensor",
      "description": "Senses environmental temperature",
      "@type": [
        "OnOffSwitch",
        "TemperatureSensor",
        "Sensor"
      ],
      "properties": {
        "power": {
          "@type": "OnOffProperty",
          "type": "boolean",
          "value": true
        },
        "temp": {
          "@type": "TemperatureProperty",
```

```
          "type": "number",
          "value": 20
        }
      }
    }
  ],
  "entities": [
    {
      "name": "Heater",
      "type": [
        "Heater",
        "Interactive"
      ],
      "thing": "electric-heater",
      "icon": "f0a0",
      "color": "#1E90FF",
      "position": {
        "x": 3,
        "y": 3
      },
      "properties": [
        {
          "id": "temp",
          "absorptionRate": {
            "type": "NUMBER",
            "value": 0,
            "unit": "H"
          },
          "diffusionRate": {
            "type": "NUMBER",
            "value": 1,
            "unit": "H"
          }
        }
      ],
      "state": []
    },
    {
      "name": "Motion Sensor",
      "type": [
```

```
        "MotionSensor",
        "Interactive"
      ],
      "thing": "motion-sensor",
      "icon": "f03d",
      "color": "#1E90FF",
      "position": {
        "x": 4,
        "y": 4
      },
      "properties": [
        {
          "id": "temp",
          "absorptionRate": {
            "type": "NUMBER",
            "value": 1,
            "unit": "H"
          },
          "diffusionRate": {
            "type": "NUMBER",
            "value": 0.5,
            "unit": "H"
          }
        }
      ],
      "state": []
    },
    {
      "name": "Temperature Sensor",
      "type": [
        "TemperatureSensor",
        "Interactive"
      ],
      "thing": "temperature-sensor",
      "icon": "f0a0",
      "color": "#1E90FF",
      "position": {
        "x": 4,
        "y": 3
      },
```

```
      "properties": [
        {
          "id": "temp",
          "absorptionRate": {
            "type": "NUMBER",
            "value": 1,
            "unit": "M"
          },
          "diffusionRate": {
            "type": "NUMBER",
            "value": 0.5,
            "unit": "H"
          }
        }
      ],
      "state": []
    }
  ],
  "activities": [
    {
      "name": "sleep",
      "entityGroup": "Beds",
      "duration": {
        "value": 1,
        "unit": "H"
      },
      "effects": [
        {
          "necessity": "comfort",
          "value": 5
        }
      ]
    },
    {
      "name": "eat",
      "duration": {
        "value": 1,
        "unit": "H"
      },
      "entityGroup": "Tables",
```

```
      "effects": [
        {
          "necessity": "hunger",
          "value": 5
        }
      ],
      "dependencies": [
        "getFood"
      ]
    },
    {
      "name": "getFood",
      "duration": {
        "value": 1,
        "unit": "M"
      },
      "entityGroup": "Fridges",
      "preconditions": [
        {
          "property": "open",
          "value": true,
          "type": "BOOLEAN"
        }
      ],
      "postconditions": [
        {
          "property": "open",
          "value": false,
          "type": "BOOLEAN"
        }
      ],
      "effects": [],
      "dependencies": []
    }
  ]
}
```

Listing C.1: Scenario 1

C.2 Scenario 2

```
{
  "rules": [
    {
      "name": "blinds",
      "enabled": true,
      "trigger": {
        "type": "TimeTrigger",
        "conditions": [
          {
            "any": [
              {
                "fact": "hour",
                "operator": "greaterThan",
                "value": 21
              },
              {
                "fact": "hour",
                "operator": "lessThan",
                "value": 2
              }
            ]
          }
        ],
        "onValue": true,
        "label": "Close blinds during the night time, open them otherwise"
      },
      "effect": {
        "effects": [
          {
            "property": {
              "type": "boolean",
              "thingId": "blind1",
              "id": "open"
            },
            "type": "PulseEffect",
            "value": false,
            "label": "effect"
```

```
        },
        {
          "property": {
            "type": "boolean",
            "thingId": "blind2",
            "id": "open"
          },
          "type": "PulseEffect",
          "value": false,
          "label": "effect"
        }
      ],
      "label": "Close blinds",
      "type": "MultiEffect"
    }
  },
  {
    "name": "movement",
    "enabled": true,
    "trigger": {
      "property": {
        "type": "boolean",
        "id": "motion",
        "thingId": "motion-sensor"
      },
      "type": "BooleanTrigger",
      "onValue": true,
      "label": "Senses movement"
    },
    "effect": {
      "property": {
        "type": "boolean",
        "thingId": "light-sensor",
        "id": "power"
      },
      "type": "SetEffect",
      "value": true,
      "label": "effect"
    }
  },
```

```
    {
      "name": "fireplace-eh",
      "enabled": true,
      "trigger": {
        "property": {
          "type": "boolean",
          "id": "power",
          "thingId": "fireplace"
        },
        "type": "BooleanTrigger",
        "onValue": true,
        "label": "Effect when the fireplace is active"
      },
      "effect": {
        "property": {
          "type": "boolean",
          "thingId": "electric-heater",
          "id": "power"
        },
        "type": "PulseEffect",
        "value": false,
        "label": "Effect when the fireplace is active; sets to
default when it is not"
      }
    },
    {
      "name": "light",
      "enabled": true,
      "trigger": {
        "property": {
          "type": "number",
          "id": "luminosity",
          "thingId": "light-sensor"
        },
        "type": "LevelTrigger",
        "levelType": "GREATER",
        "value": 50,
        "label": "Turns off lights"
      },
      "effect": {
```

```
        "property": {
          "type": "boolean",
          "thingId": "light-sensor",
          "id": "power"
        },
        "type": "PulseEffect",
        "value": false,
        "label": "effect"
      }
    },
    {
      "name": "pressure",
      "enabled": true,
      "trigger": {
        "op": "AND",
        "triggers": [
          {
            "property": {
              "type": "number",
              "id": "windSpeed",
              "thingId": "blind1"
            },
            "type": "LevelTrigger",
            "levelType": "GREATER",
            "value": 20,
            "label": "Check wind speed"
          },
          {
            "property": {
              "type": "number",
              "id": "windSpeed",
              "thingId": "blind2"
            },
            "type": "LevelTrigger",
            "levelType": "GREATER",
            "value": 20,
            "label": "Check wind speed"
          }
        ],
        "label": "Check wind speed",
```

```
        "type": "MultiTrigger"
      },
      "effect": {
        "effects": [
          {
            "property": {
              "type": "boolean",
              "thingId": "blind1",
              "id": "open"
            },
            "type": "PulseEffect",
            "value": false,
            "label": "effect"
          },
          {
            "property": {
              "type": "boolean",
              "thingId": "blind2",
              "id": "open"
            },
            "type": "PulseEffect",
            "value": false,
            "label": "effect"
          }
        ],
        "label": "Close blinds",
        "type": "MultiEffect"
      }
    }
  ]
}
```

Listing C.2: Scenario 2

C.3 Scenario 3

```
{
  "activities": [
    {
      "name": "sleep",
```

```
      "entityGroup": "Beds",
      "duration": {
        "value": 1,
        "unit": "H"
      },
      "effects": [
        {
          "necessity": "comfort",
          "value": 5
        }
      ]
    },
    {
      "name": "eat",
      "duration": {
        "value": 1,
        "unit": "H"
      },
      "entityGroup": "Tables",
      "effects": [
        {
          "necessity": "hunger",
          "value": 5
        }
      ],
      "dependencies": [
        "getFood"
      ]
    },
    {
      "name": "getFood",
      "duration": {
        "value": 1,
        "unit": "M"
      },
      "entityGroup": "Refrigerators",
      "preconditions": [
        {
          "property": "open",
          "value": true,
```

```
          "type": "BOOLEAN"
        }
      ],
      "postconditions": [
        {
          "property": "open",
          "value": false,
          "type": "BOOLEAN"
        }
      ],
      "effects": [],
      "dependencies": []
    },
    {
      "name": "work",
      "duration": {
        "value": 3,
        "unit": "H"
      },
      "entityGroup": "Computers",
      "preconditions": [
        {
          "property": "power",
          "value": true,
          "type": "BOOLEAN"
        }
      ],
      "postconditions": [
        {
          "property": "power",
          "value": false,
          "type": "BOOLEAN"
        }
      ],
      "effects": [
        {
          "necessity": "income",
          "value": 15
        }
      ],
```

```
      "dependencies": []
    },
    {
      "name": "shower",
      "duration": {
        "value": 15,
        "unit": "M"
      },
      "entityGroup": "Showers",
      "preconditions": [],
      "postconditions": [],
      "effects": [
        {
          "necessity": "hygiene",
          "value": 50
        }
      ],
      "dependencies": []
    },
    {
      "name": "leisure",
      "duration": {
        "value": {
          "minimum": 1,
          "maximum": 3
        },
        "unit": "H"
      },
      "entityGroup": "Sofas",
      "preconditions": [],
      "postconditions": [],
      "effects": [
        {
          "necessity": "comfort",
          "value": 15
        }
      ],
      "dependencies": [
        "turnOnTV",
        "watchTV",
```

```
        "turnOffTV"
      ]
    }
  ],
  "agents": [
    {
      "name": "tiago",
      "position": {
        "x": 15,
        "y": 15
      },
      "icon": "f183",
      "color": "#6daf2b",
      "activities": [
        "work",
        "shower",
        "sleep",
        "eat",
        "getFood"
      ],
      "necessities": [
        "comfort",
        "hunger",
        "hygiene",
        "income"
      ]
    },
    {
      "name": "ana",
      "position": {
        "x": 12,
        "y": 4
      },
      "icon": "f183",
      "color": "#0000FF",
      "activities": [
        "leisure",
        "shower",
        "sleep",
        "eat",
```

```
        "getFood"
      ],
      "necessities": [
        "comfort",
        "hunger",
        "hygiene"
      ]
    },
    {
      "name": "bruno",
      "position": {
        "x": 3,
        "y": 4
      },
      "icon": "f183",
      "color": "#FF0000",
      "activities": [
        "sleep",
        "shower",
        "eat",
        "getFood"
      ],
      "necessities": [
        "comfort",
        "hunger",
        "hygiene"
      ]
    }
  ]
}
```

Listing C.3: Scenario 3