Unleashing the Future of Smart Homes: A Revelation of Cutting-Edge Distributed Architecture

Joselito Jr
joselito.mota@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Luana Martins
martins.luana@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Dhyego Tavares
dhyegocruz@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Denivan Campos
denivan.campos@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Frederico Durão
fdurao@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Cássio Prazeres
prazeres@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Maycon Peixoto
maycon.leone@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Gustavo B. Figueiredo
gustavobf@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Ivan Machado
ivan.machado@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Eduardo Almeida
eduardo.almeida@ufba.br
Federal University of Bahia
Salvador, Bahia, Brazil

Abstract

Recent discussions have delved extensively into Smart Homes, focusing on constructing and integrating services into an architecture capable of supporting the daily routine of the homes of several users and providing a stable operation. When artificial intelligence is added to a development architecture with devices that interface between man and machine, creating a system that operates in conjunction with users becomes a real challenge. How the system may or may not behave in the day-to-day lives of users becomes the genuine concern of developers. This brings challenges to be overcome and decisions to be made. Implementation decisions, division and creation of modules, communications between applications, user actions, communication interfaces, server response times, and other issues arise amidst all the complexity of developing for the real world. Observing the current scenario, we present our proposal for a distributed Smart Home architecture integrated with a third-party SAAS-based cloud service with an extensive catalog of smart devices used in a smart home environment. Our architecture obtains data about devices from users' homes registered in a third-party cloud, and uses Artificial Intelligence to train a model using the user's routine based on the behavior and use of devices in the house. The system provides recommendations sent directly to the smartphone or smartwatch to help with user comfort or to reduce energy consumption. This work presents the system architecture, technologies, and communications between services. Ultimately, we list the lessons learned in architectural design, solution coding, module integration, communication with smartphones and smartwatches, and working with intelligent physical devices in the user's environment.

CCS Concepts

• **Software and its engineering → Software system structures**.

Keywords

smart homes, software architecture, software design, machine learning applied, lessons learned

1 Introduction

State-of-the-art discussions are that the residential future lies in autonomous houses [3, 12]. Remote control of devices, even via smartphones, is now a reality, but this is nothing more than automation or direct action by the user. Although devices are connected to the internet, people must still control their actions [13]. The functionality closest to giving intelligence to devices, such as lamps, plugs, remote controls, and others, is pre-defined modes and automation that the user configures [1]. Nevertheless, this action becomes tiring if users have many devices (smart plugs, smart-bulb, and others).

In addition to devices still needing external actions to function, there is no documented structure for inserting intelligence into devices, such as smart recommendations for previous user actions. In other words, smart devices still need to be fully intelligent and autonomous; they need user input to work.

Architecture proposals exist in smart devices based in Raspberry, Arduino, and others [2] [14]. The situation is different for smart devices sold in retailers that typical users widely purchase. The difficulty in handling retail devices lies in integrating third-party partners with the rights and control of these devices. Therefore, an intelligent solution must follow the performance and data reception standards imposed by the manufacturer when they exist. Additionally, adding intelligence to devices requires mastery of artificial intelligence to bring life to devices throughout the home. Adding artificial intelligence in the development architecture with devices that interface between man and machine, creating a system that operates in conjunction with users becomes a real challenge because every restriction and variable must be counted.

How the system may or may not behave in users' daily lives becomes a genuine concern of developers and influences decision-making in code, creation of architecture, communications between applications, user actions, communication interfaces, server response times, and other issues that arise in the complexity of developing for the real world.

Regarding the process of developing an architecture that accommodates the use of real-world devices and brings intelligence to

devices and practicality to the user's daily life, this study addresses the complexities and the decision-making involved in creating a smart home architecture using artificial intelligence. Our study shows a multi-platform distributed architecture for smart homes that recommends actions and user interactions with the devices in the environment based on machine learning (smart home, smart spaces, smart malls, and others). Our solution is used in a real-world environment deployed on the university campus, but smart homes can be the system's target. The development was carried out over two years. To conduct the study report, we raised the RQs to be answered as lessons learned:

RQ1: How were the standardization and project design decisions made to drive the development of a reliable architecture always available to the user in smart homes? The study presents how decisions were made about the standardizations used in the architecture from the design phase to the integration of modules. In addition to standardizing message exchanges, we highlight the communication factor between teams from different modules to construct the entire system successfully. We also present some aspects of real-world smart device technologies and how it was necessary to understand technologies and the limitations that each one brings to developing a smart home architecture.

RQ2: How was the technology implemented and transferred from the development environment to the real world? The decisions made to integrate systems and correctly initialize servers were part of this question. We use state-of-the-art dev-ops technologies applied to the smart home architecture modules built in this article. This study discusses the scalability of containers and Kubernetes virtualization for deploying and technologically transferring construction artifacts.

RQ3: What efforts were taken to mitigate infrastructure issues in an intelligent home-based architecture? We also present some issues and lessons learned to mitigate infrastructure problems that a smart home architecture can bring when deployed from the development phase to the real world. We face physical problems, networks, and even physical limitations regarding devices. The actions taken to overcome the situations were crucial for conducting and continuing the development process and are described in this article.

Our smart home architecture handles communications between third-party services, APIs, Apache Pulsar, and sending and receiving data from the home to Smartphones and Smartwatches. Our architecture modules were developed using REST API technology to provide users with personalized recommendations based on interaction with smart devices. Action recommendations are sent directly to the smartphone or the user's wrist on a Smartwatch, in addition to manipulating and receiving data about the home's efficiency and energy consumption. There is also room for administrators to monitor the system based on usage, quantity, efficiency, and consumption of homes and devices. We also share lessons learned over the two years of development, the actions that went right and what went wrong, and the paths and recommendations we followed along the project.

2 An overview of the smart home literature and their proposals

Considerable effort has been dedicated to investigating artificial intelligence for smart homes, focusing on cybersecurity, device management, user interaction, energy efficiency, and health care.

A summarized state-of-the-art, comprehensive literature review from 2011 through 2019 made by Guo et al [3]. The studies classified current work into six clusters: activity recognition, data processing, decision-making, image recognition, prediction-making, and voice recognition. Others studies reports device management by decision-making and image recognition. Studies involving health care employ activity recognition to perform those actions. Finally, the studies about energy management demonstrate the use of prediction-makin' to perform those actions.

A literature review from 2011 through 2019 to investigate the application of the IoT in homes to make them automated and intelligent is the contribution of Sepasgozar et al. [12]. The authors analyzed the frequency of words in the papers and derived scenes for the main topics studied in the literature. Building upon this foundation, our proposed framework stands out as a complete solution for smart homes. By leveraging activity recognition, decision-making, and prediction-making, our framework addresses critical aspects of device management, energy efficiency, and intelligent interaction.

Device management. Crisnapati et al. [2] introduced Rudas, a home automation system that enables remote monitoring and management through a web interface. Equipped with artificial intelligence, the system autonomously controls lighting and temperature while allowing manual device control for comprehensive access. Differently, Jivani et al. [6] proposed a voice-and AI-based centralized management system that will enable users to control domestic appliances and services with voice and also makes decisions on the end user's behalf, such as monitoring, improving comfort, and delivering required information whenever needed. Our work focuses on smartphones and smartwatches, interactive notifications learned from users' historical actions on devices. The user also decides which actions the recommendations generated are valid for everyday use.

Intelligent interaction. Ospan et al. [10] developed a Virtual Assistant that is used as a control interface of the Smart Home environment. Specifically, the system is constructed to give multiple users the ability to control appliances by voice or text commands. The authors used artificial neural networks to classify user inputs and create a natural dialogue. Further, a set of random double-blinded evaluation tests were performed with generally positive results in terms of interface justification. Su et al. [14] presented a gesture recognition algorithm using an accelerometer and gyroscope based on a neural network. In addition, they showcased the application in a mobile game for hand rehabilitation training of patients after hand injury surgery and in an intelligent socket to manage household electricity. Our work differs by using user activity recognition; for example, a turn-on light action triggers a new recommendation for the user instead of voice/image recognition.

Energy efficiency. Jithish et al. [5] investigated the impact of environmental factors on home energy consumption for developing efficient energy management policies for smart homes. In addition,

the authors developed an adaptable rule-based demand estimation system from artificial neural networks using fuzzy logic to predict the daily home electricity demand. Paul et al. [11] presented an approach combining IoT and machine learning mechanisms to predict smart builds indoor temperature. The prediction of indoor temperature reduces the overall energy consumption of the building, accounting for heating and cooling demand, by automatically controlling the high energy-consuming devices over the network. Our work employs smart devices' carbon footprints and energy consumption to recommend actions to reduce the consumption of devices by the user. The recommendations apply in two ways: by the user's feedback or based on previous users' actions. This applied action of the recommendation is based on the consumption of the device that will try to reduce the electrical energy consumption of the smart environment.

More general frameworks. More recently, Manandhar et al. [8] proposed and built a framework, Hєlion, that leverages natural home automation scenarios using trigger-action user interfaces. The paper empirically reported the naturalness hypothesis with 273 routines collected from 40 users. The paper presented the usefulness of Hєlion by generating 17 security/safety policies with minimal effort. Later, Mandal et al. [9] presented the implementation of HєlionHA, an extension of a popular home assistant with Hєlion. The paper contributed to the practical applicability of the framework by providing realistic scenarios for testing the security and safety of smart homes. Our work proposes a comfortable and energy-efficient scenario, delivering recommendations to users to perform those actions in an smart environment.

3 Smart Home Architecture

Our Smart Home Architecture is designed with the objectives of facilitating seamless communication between modules without overloading requests, ensuring efficient interoperability of services with high availability and minimal failures, and establishing communication with third-party devices in the Cloud, as seen in Figure 1. The following section presents the components of our architecture and delineates the communication channels with third-party cloud services.

As the project is subject to confidentiality contractual clauses, we cannot make data such as source code and other crucial information regarding third-party client details available.

3.1 Third-party Partner

An aspect to consider is the service structure on the devices we use to get the data from third-party clients. Third-party clients have a cloud based on the SaaS model, where a third-party infrastructure provides an abstraction over users, houses, rooms, devices, and other structures. Thus, the third-party partner has a range of devices for residential environments, such as lamps, socket plugs, and others. Users purchase these products at regular retail stores and connect the devices to this third-party cloud that publishes data that can be accessed via API or Apache Pulsar[1]. Thus, our system uses the communication mechanisms mentioned above to be able to listen and act on those smart devices and environments.

[1]https://pulsar.apache.org/

3.2 Core Module

The Core module is responsible for obtaining data about users, homes, rooms, devices, and intelligent products from the third-party cloud API. This module works as an orchestrator in our architecture, whereas the other modules send and receive data from the Core. Furthermore, the Core is responsible for copying the necessary data indexes and structures from the Smart Homes in the third-party cloud service.

3.3 Smart Module

A specific module in our smart home architecture is responsible for receiving data from the environment and devices, adjusting the data, and training using the LSTM (Long Short-Term Memory) [4], a recurrent neural network (RNN) architecture that uses value intervals, to provide recommendations to the user based on routine actions performed with the devices in the home. The smart module receives data about the device, as shown in Figure 2, its status (ON/OFF), the time of action, the room of the device, and the house in which the action occurred. From this data, the Smart Module uses recurring data to learn the users' usage routine, considering the actions performed in the house. These recommendations are sent to the Core, which forwards the recommendation message to the next module, Push Notification Module. The learning flow of the Smart module is described in Figure 4.

3.4 Notification Module

This module is responsible for receiving Core recommendations generated by the Smart Module, decoding them into text for the user, and passing them on to smartphones and smartwatches. Devices constantly consult the endpoints of this module to receive recommendations. Recommendations that remain for more than 15 seconds without responses are automatically discarded for the user. The Push Notification module is used externally to give the system more scalability, as several requests are made by smartphones or smartwatches.

3.5 Graphics and Wearables Module

The Graphics and Wearables module calculates user, home, rooms, and device data to send to smartphones and smartwatches based on user usage. The data calculated refers to energy consumption, number of devices, rooms, and recommendations numbers. Following the logic of Push Notification, this module anticipates numerous smartphone and smartwatch requests from numerous house users by allocating specific endpoints to access data. To prevent overloading Core, we relocated it and established a dedicated server for transmitting data to users about their devices and homes. This module performs analyses, including those for energy efficiency and energy consumption in the user's home, data on devices and interactions, information on recommendations generated for home residents, and other details on user permissions.

3.6 Smartphone Application

The application, programmed on the Android platform, utilizes third-party client libraries to manipulate devices and abstractions related to houses and rooms. It interacts with the Push Notification

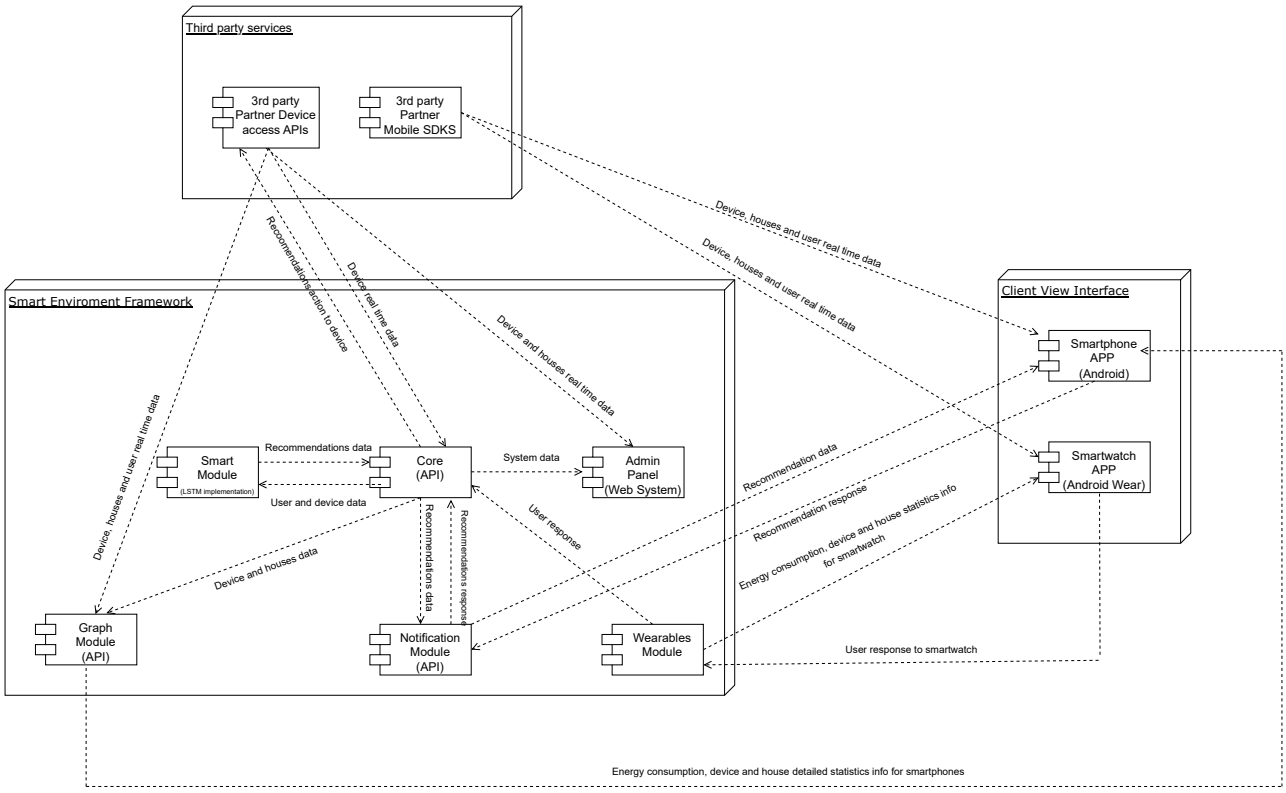

Figure 1: UML component diagram of our Smart Home Architecture

server to receive and send recommendation responses. The application also connects with Mobile Provider Service to fetch calculated data on energy efficiency, energy consumption, and system and device usage. Users can view only the data related to their homes and devices in this application, which offers a comprehensive view of the home, as shown in Figure 3.

3.7 Wearables Application

The application, programmed on the Android Wear platform, interacts with the Push Notification server to receive and send recommendation responses. The application also connects with the Wearables Service to obtain data about the home and manipulate smart devices.

3.8 Administrator Panel

The Administrator Panel shows the administrator all information about the houses and devices with interactive charts and information tables. The application is only accessible by authorized people and shows information about other modules' data, like recommendations. The graphs and maps concentrate data on users' homes.

The upcoming section discusses module communication, which is crucial, as not all applications communicate directly for security considerations or to mitigate potential module overloads. Consequently, the following topics will delve into the connections, data exchanges, and messaging protocols, providing reasoned justifications for sending and receiving data within this context.

4 Communications between modules in Smart Home Architecture

To comprehensively understand the architecture of our smart home system, it is essential to examine the communication between modules and their functions. With this, we summarize communications between modules based on the flow of user actions to perform a particular action in the system. Thus, everything starts with the user acting on a smart device, triggering all the learning of our system, as shown in Figure 4.

4.1 Unlocking It All: Understanding User Interaction with the Device

The system's first trigger happens when the user acts as our application, which is based on the Android platform and has specific SDKs from our third-party partner.

The third-party partner SDKs have methods encapsulated for connecting and manipulating devices and creating houses, rooms, and users. Thus, the Android application we created imports these methods from the third-party partner SDK for user action. It is essential to highlight that using the third-party partner SDK libraries, the action to the device performed by the user is sent directly to our

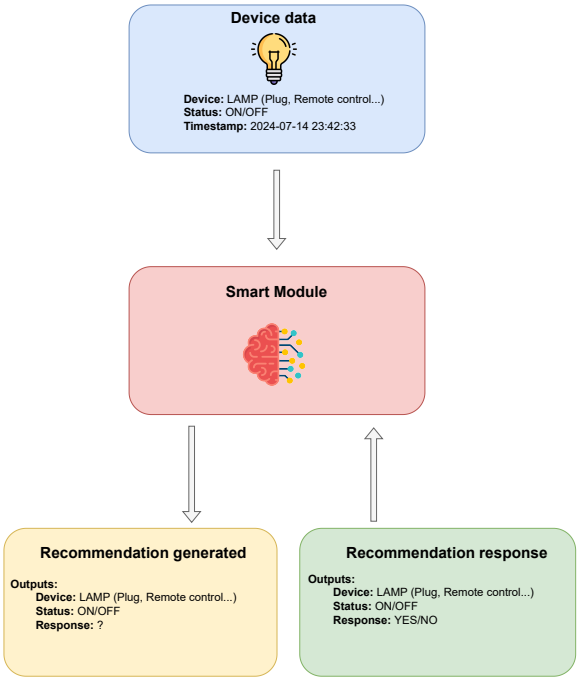

Figure 2: The data is passed to the Smart Module, which uses an LSTM for training, and a recommendation is generated for the user. After the recommendation is generated, it is sent to the Core, which is responsible for making it available to the notification modules, which trigger the message to all devices (Smartphones and Smartwatches) of the users in the house.

third-party partner's SaaS-based cloud over an encrypted connection, which sends it to the smart device connected to the network.

For example, in a connected, smart light bulb, the user turns off the light bulb with our application, which will use our third-party partner's method to send the action to the cloud, which forwards the request to turn off the light bulb.

4.2 Capturing device actions and generating recommendations

The capture of user action on the device comes after the data arrives in the third-party partner's cloud. The third-party partner provides two options for querying the data, one via API and the other via Apache Pulsar connections.

After receiving the data, our Core module, a REST API service, observes the action, copies the data, and adjusts it to send it to the Smart Module to learn from it.

When data is sent, the Smart Module, which uses machine learning to create recommendations, learns from the user's behavior and makes a recommendation that best suits the user based on the trained mechanisms. The recommendation created for the Smart Module is sent to the Core module, which processes the information and passes it on to the Push Notification server to forward to the recipient.

In the previous example, when turning off the lamp, the Core module captures the action, passing it on to the Smart Module that is learned from the user. Based on previous learning, the Smart Module may or may not recommend an action. Suppose a recommendation to turn on the lamp is created. In that case, the recommendation is immediately sent to the Core, which sends it to the Push Notification and delivers it to the sender correctly and legibly. The recommendation is sent via JSON, in which the structure lists the actions based on key and value.

4.3 Sending generated recommendations directly to the user

Once the recommendation is created, the Push Notification server receives the recommendation, which is still unreadable to the user. The function of the Push Notification module is to decode the recommendation into a common reading language and pass it on to the user. For this, the module contains dictionaries to transform the command from JSON into the current language and save it in the database on the waiting list.

When the recommendation arrives to the user, the smartphone or smartwatch observes the recommendations endpoint, checks whether a new recommendation is registered, and shows it to the user on the smartphone. Recommendations are sent via JSON and are only valid for 15 seconds, with old ones removed from the recommendation list. In the previous case, the user will receive a recommendation to turn on the lamp in an alert or can consult the list of recommendations on their smartphone or smartwatch.

4.4 User response to recommendation and action across devices

The user responds to the recommendation via their smartphone or smartwatch, and then the smartphone or smartwatch's mobile application directly sends the user's response to the Core module with JSON.

The Core module sends the response data to the Smart Module to learn from the user's feedback and, at the same time, acts on the device by sending the command to the smart device.

When recommending to turn on the lamp, the mobile application sends the recommendation response to the Core. The Core module communicates with the third-party cloud via the available endpoint, passing the device identification and applying the recommendation action to the device.

However, if the user does not respond, the Smart Module will decide, based on previous learning, what action to take with the recommendation, whether to accept or decline.

4.5 Getting system-wide metrics for admin

The administrator panel is an entirely separate and isolated system. The panel queries the Push Notification modules' data to find the recommendation data. A JSON is sent with the list of recommendations and the user's responses.

In addition to consulting the Push Notification server, the Administrator Panel queries the third-party partner's Cloud data. Thus, it searches for users, houses, rooms, and devices to create metrics

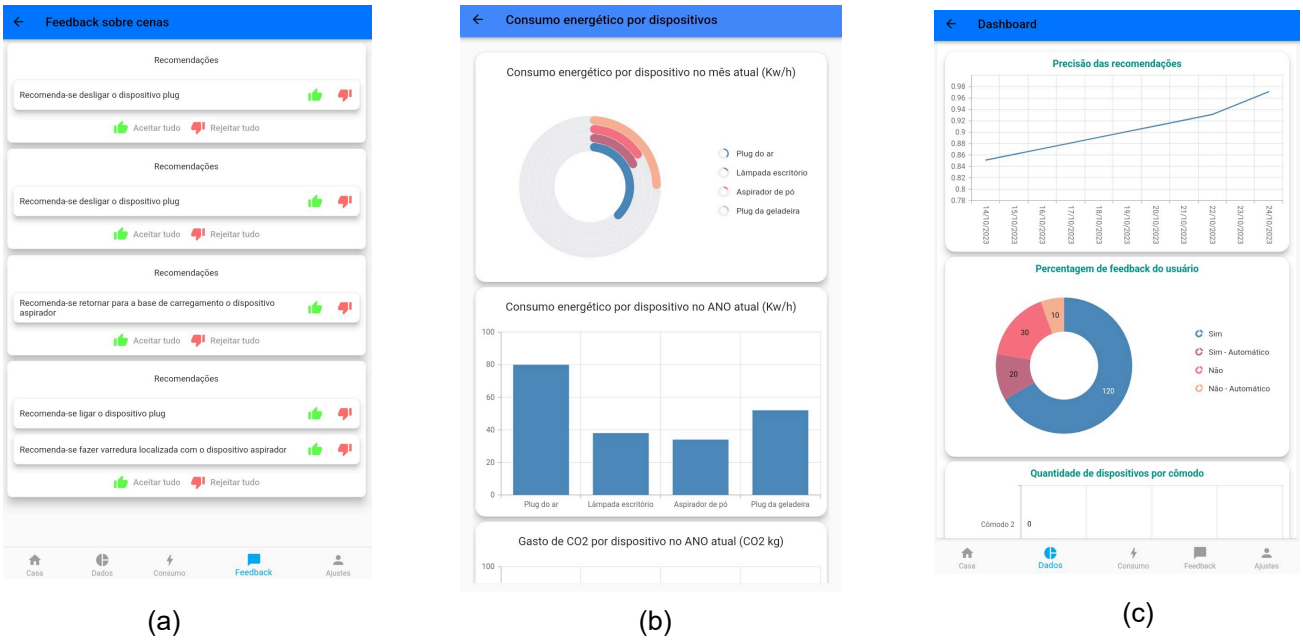

Figure 3: Application screens developed in our study. (a) Screen of previous recommendations for the device. The screen consists of recommendations received from the modules to accept or reject user action. (b) Screen showing energy consumption data for home devices. (c) Screen on data from prediction of recommendations in the house.

on the number of houses, devices, which devices are most used, energy consumption and efficiency, status of house devices, data on the responses to the recommendations generated, and accuracy of the recommendation system.

4.6 Obtaining house metrics to send to the user

The graphics module is focused on calculating the metrics for each house in the system and provides the data to its users. The server calculates the number of devices, users, rooms, and houses and receives calculated efficiency and energy consumption data from each house. The mobile application queries the graph server to receive the JSON information necessary to plot graphs on previously listed metrics.

The Wearables module calculates metrics for graphics and manipulates smartwatch devices. The smartwatch constantly queries the Wearables server, which sends and receives JSONs.

Both servers were created separately to receive many connections and requests and avoid bottlenecks in the rest of the modules that handle device actions. This way, the server can manage many Smartphone and smartwatch requests.

5 Architecture originality and advancements

In this section, we present some novelty topics that our system and architecture bring to the smart home scenario. Our module organization, backed by the use of cutting-edge technology and implementations, is the driving force behind the entire process of receiving a recommendation and executing a device action in the user's home.

5.1 Why do we propose it as actually being a cutting-Edge distributed architecture?

By leveraging activity recognition, decision-making, and prediction-making, our framework addresses critical aspects of device management, energy efficiency, and intelligent interaction. Besides, our work focuses on smartphones and smartwatches that are the most sophisticated and current on the market (Android, iOS, and Android Wear) with interactive notifications learned from users' historical data on actions in house devices. All these devices with different platforms and user interfaces are managed by a set of modules that communicate with each other, exchanging data so that the action recommendation, decision-making, action, and consequences of actions are captured and applied to the devices using machine learning with this historical information from the user and the devices inserted in the environment. Likewise, the architecture of the intelligent system can be adapted to different devices available and perform comfort or energy consumption modes.

Also, the architecture originality characteristics of our system that impact both the user and the entire system stack are:

(a) Architecture is integrated with an artificial intelligence agent that determines the behavior of environments: Our architecture has an intelligence module called Smart Module, composed of an LSTM (Long Short-Term Memory) [4], a recurrent neural network (RNN) architecture that uses value intervals. In this case, the routine usage of users' home devices is recorded and used for training to generate new recommendations for actions in the home based on comfort or energy savings. The choice of comfort

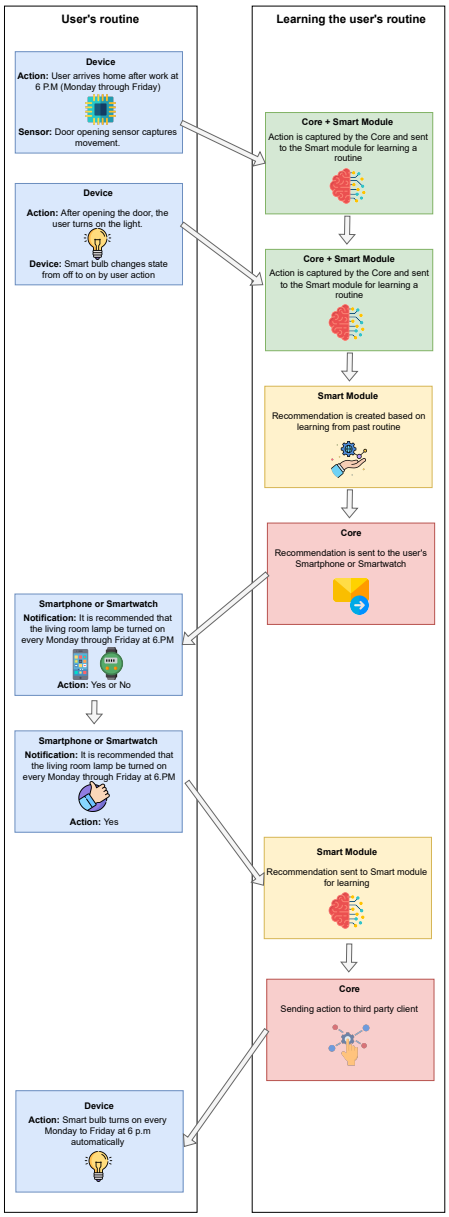

Figure 4: Recommendation workflow communication

or energy savings mode is up to the user, and the recommendations are delivered based on the chosen behavior.

(b) Recommendations for device actions are generated based on user interaction with smart devices in the environment: The user's interaction with the smart environment and available devices determines the generation of recommendations for a given device. Thus, recommendations are also created based on the profile that the user wants in their smart environment (comfort or energy savings). It is up to the user to accept the activity's

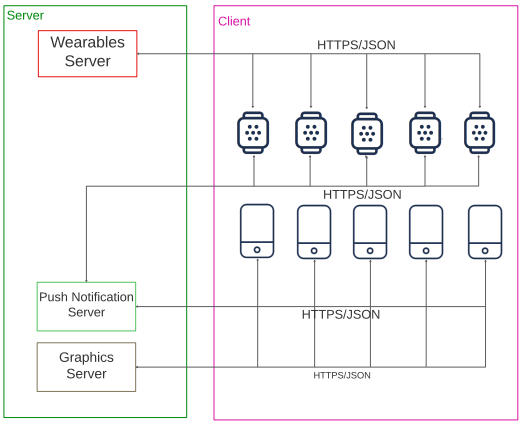

Figure 5: Smartphone and Smartwatch communication

recurrence or refuse to perform a specific action. The home learns from the resident's schedule and habits and increasingly improves the quality of the recommendations.

(c) Real environments and devices used with respective user routines: The architecture and system are deployed on the university campus using real devices from third-party manufacturers. A lab was fully modified with smart devices and sensors to capture user actions and deliver action recommendations based on lab users' routines.

(d) Possibility of integrating multiple intelligent systems outside the scope of the original architecture: The system architecture supports adding new devices and interfaces. For example, virtual assistants such as Alexa, Google Home, and Siri can integrate data from modules such as graphs and notifications into the original architecture without further modifications. In other words, we create multiple modules as an interface for integrating other devices into the current architecture.

(e) The system and the architecture allow container orchestration and an application management system to deploy the newest platforms: The system's architecture and use of Docker containers make it easy to move and integrate across different environments. By leveraging both Docker and Kubernetes, we gain a range of advantages that significantly improve the system's functionality and reliability. Kubernetes acts as the conductor, ensuring smooth scaling and high availability through features like load balancing and automatic failover. Docker containers, on the other hand, package applications with their dependencies, guaranteeing consistent behavior regardless of the environment. Additionally, Docker contributes to resource efficiency by sharing the underlying operating system kernel with other containers, requiring less overhead compared to traditional virtual machines.

(f) Possibility of configuring duplication and redirection of services in case of failure in any module: Using Kubernetes allows us to ensure system reliability and resilience through its advanced failover and redundancy mechanisms. Kubernetes enables the configuration of multiple replicas of a service across different nodes, which helps distribute the load and provides fault tolerance. In the event of a failure in any module or instance, Kubernetes can

automatically redirect traffic to the remaining healthy instances, ensuring uninterrupted service supporting self-healing capabilities by continuously monitoring the health of running services.

(g) Independence between services offered: Since the architecture is distributed and different modules perform separate feature processing, service availability is unaffected when a module fails. For example, suppose the graphics module fails, and the module's replication system is unavailable in time. The user can still access services from other modules, such as device recommendations and actions. Functionalities are not completely compromised, and the user can perform other actions that communicate with available modules in the architecture. This method of distributing services not only avoids causing the entire system to go offline, but the behavior of a module does not influence the entire functioning of the services.

(h) System monitoring through web administration panel of system resources and data: The system administrator has access to a panel, a WEB application, with all data about the devices and the Smart Module, data about generated recommendations, Smart Module performance in the system, and other metrics related to the quality of recommendations and data about the platform devices.

(i) The system acts automatically on the user's devices without the user having to take extra steps: Once the machine learning has been trained and sent an action recommendation to a device and the user through the smartphone and smartwatch interfaces has approved the request, the action on the device and in the environment is performed without any intervention or assistance from the user on the smart device (plug, lamp, and others). The system will perform the actions automatically from now on until the user stops the activity or receives and accepts another recommendation to stop the action. The system gives intelligence to the devices using the modules, messages, and interfaces behind the system architecture.

6 Lessons Learned

In this section, we present the lessons learned from the entire architecture development process, from conception to the implementation phase in the natural environment. Here, we list some design decisions we used in our architecture and other directions to construct the smart home system architecture application. Each lesson learned answers the challenges in RQ1, RQ2, and RQ3.

RQ1: How were the standardization and project design decisions made to drive the development of a reliable architecture always available to the user in smart homes?

(1) Standardize the exchange of information between modules

At the beginning of the architecture development, the team raised a concern: how would they rapidly integrate modules and functionalities to minimize communication problems when finishing the modules' development?

When integrating third-party software, compatibility problems in the exchange of information may arise that lead to delays in the development process. The modules use REST API technology, and each module would be an API that receives and provides data. The decision made by the development team to minimize possible incompatibilities was to standardize all types of communication between modules. Thus, the developer's team defined all communication standards before the coding phase. Artifacts such as JSON message exchanges, access endpoints, and sending data were standardized and fixed. Each developer had to follow both the data to be exchanged and the naming rules, data order, and list of endpoints that each module needed to perform a specific action throughout the environment. For example, the recommendations generated by the Smart Module should be sent from the Core to the Push Notification Server to reach the user's smartphone. The message responsible for communicating between the Core and the Push Notification Server to send the recommendations was set as a standard just before the development phase. Thus, errors were minimized by integrating the two modules developed by different teams, and the integration was carried out in less time than expected.

(2) The essence of communication between developers of different modules

Even with the standardization of messages, some paths had to be changed, whether due to technology limitations, better use of code, or refactoring. Many of these changes were sometimes not passed on to the other team when integrating the modules, and they were surprised by a new form that was outside the agreed standard. Even so, the integrations that followed the standardization were less problematic than those carried out when some communication was not followed as previously defined. Evidence of delayed integration, when the standard deviates from the agreed one, confirms that the standardization mentioned above shortens the path of software development. Not only that, but the team's communication when standards change is of fundamental importance and can impact the coding time, integration, and delivery of the solution as a whole to the client.

(3) Understanding the limitations of each technology used in the project

Mobile technology had a limitation on the size of the message sent to the core. The current defect slowed down the development and integration process, as the message had to be broken into several parts, altering the planning that had previously been made for communication between modules and applications. With the mitigation of technology limitations carried out at the beginning of development, the defect would not have existed, and the need to change the messages sent would not have occurred. Despite the defect being found and corrected, there were no delays in delivery. Still, on another occasion, this could lead to drastic changes in the architecture and a delay in delivery deadlines. To avoid more significant issues in future development, it is necessary to have a thorough, deep knowledge of the technology and ask if it is the best choice for the scope of the problem.

(4) Working with data from a third-party partner, understand all data flows, means of communication, and operation of APIs

In our smart home architecture, we deal with a third-party client who provides the data and devices in the client's home. All data from devices is sent to a third-party cloud that makes the data available via an API or Apache Pulsar. In the development process, we deal with restrictions of the third-party partner in obtaining and acting

on the devices. To overcome the limitations, studying the customer documentation and planning to fit the scope of the application was necessary and crucial to begin implementing the system and its architecture. Investigating and testing the API, message exchanges, and devices were essential to building the message exchange and structuring the databases and logic of the existing modules. In addition, taking time from system development to understand the API's and technologies that will be integrated into the system is essential for the system development process. The architecture that meets the user's needs well—in addition to avoiding certain misdirection's in the project that could delay the delivery of the software.

(5) Always remember the maximum availability in crucial modules to provide the best service to the user

The scope of the application deals with everyday, real-life user actions. They are devices that operate in homes day and night, performing all types of operations to the user's liking. Services must always be active for user recommendations to be delivered via many smartphones and smartwatches. Thus, server overloads could not happen, as user needs are instantaneous. For example, a recommendation must be generated, sent to the user, and applied correctly. Therefore, the decision to have a decentralized distributed architecture where different REST APIs perform functions motivated the project's construction. If a Graphics server has a problem, the recommendation will usually arrive at the devices. The modules can be replicated as it is an independent REST API independent of the others. For example, if the Push Notification server stops working, another instance is created and takes control.

(6) When implementing a new functionality, consider system action usability in the user house

Recognizing system requirements is a crucial activity for the success of the application. This rule is also a guide to success in the smart home business. Understanding user demands and how a given module can influence positively and negatively daily helps rule out excellent features and others that could be better. In our smart home example, the Push Notification server sends messages to the user when a recommendation is created, sending it to the user's smartphone or smartwatch if they allow it. The choice for mobile development in the application and consequently inserting modules that the cell phone will constantly consume came from users' high use of smartphones. Thus, our system communicates directly with the user using our mobile application, whether on a smartphone or smartwatch. In addition to instant messaging with recommendations, graphics servers provide these mobile applications with data about the home, providing comfort and helping with various processes such as energy saving and home management.

RQ2: How was the technology implemented and transferred from the development environment to the real world?

(1) Modules and Dev-ops may take longer than considered

During the architecture project, we envisioned a simple dockerization to virtualize all the software components in the initial stages to facilitate system integration later. However, as the Core expanded and processing demand surged, it became imperative to adopt Kubernetes-based virtualization for enhanced horizontal scalability and better container orchestration due to many microservices running on our Core simultaneously.

In the smart homes domain, everything needs to be connected and online. Still, during the operational development, one of our biggest obstacles was running all our services on local servers without using pre-configured services such as AWS or Azure. Starting a local cluster of servers that supports Kubernetes and shared volumes that run microservices simultaneously requires much more time and knowledge of Kubernetes and server configurations than we expected.

RQ3: What efforts were taken to mitigate infrastructure issues in an intelligent home-based architecture?

(1) Additional challenges?

In the scope of smart homes, we have many technologies and abstractions that the team is in the development phase (third-party APIs, devices, infrastructure, and others). Our work has several layers of action as physical devices, control and obtaining data with third-party services, programming modules, and our APIs, mobile smartphone applications, and smartwatches. With this structure of many artifacts to deal with, some unplanned problems and delays may appear, which in most cases are silent to detect but cause many severe problems in the functioning of the final product. Some of the direct infrastructure problems were with requests from third parties, where we had to deal with the limitations encountered in our systems or sometimes follow another plan. In particular, we had network problems and bandwidth consumption, issues with communication with third-party servers, or even our applications not operating on the devices sometimes. These problems had to be fixed during coding, and it was often necessary to mobilize the entire team to fix them. In addition, compatibility with the libraries used in the server and developer environment and the permissions delayed project implementation planning.

Therefore, one of the main lessons learned in creating a large solution with several distributed modules is that even when planning architecture, standardizing, documenting, integrating, and testing distributed modules, some extra problems will appear and must be investigated and resolved. Thus, when these extra issues arise, it is necessary to list the possible causes presented and potential solutions for mitigation, not leaving aside any doubts and always documenting the entire fact to avoid repetition.

7 Threats to validity

Internal Validity: The devices that support the architecture were selected based on the devices available in the third-party client's product line. This selection also considered the devices (sensors and actuators) most used by users in the home. The architecture decisions, recommendations, and visualizations were built and improved for the main devices on the list, leaving out less-used devices.

The construction of the architecture and the structuring of the modules were not impacted by the choice of devices but by the platforms that the architecture would target, the construction of APIs, the sending and receiving of messages from smartphones and smartwatches, and the possible addition of an external device, such as personal assistants.

External Validity: Considering that the architecture was developed for a type of smart environment, the context of use was a smart college campus, more precisely, a laboratory. The generalization of the system's use, as well as the lessons learned, may be limited to more generic cases. To overcome this limitation, our architecture uses technology from real-world devices that can be used in various environments and factors, such as homes, offices, and other environments. The devices' actions do not change depending on the environment, only the routine in which they are inserted. Therefore, the impact of the system in other smart places is impacted by the routine of its users in the environment, not the influence of the environment itself.

Using a third-party framework to obtain data and use devices, the lessons learned and system communications are applied to the context of the device manufacturer. However, the manufacturer's standards, such as Apache Pulsar and API communications, are universally known in the IoT world. Thus, communications between modules, implementations, and lessons learned can present results consistent with the implementation in current smart devices.

Construct Validity: The listed difficulties and lessons learned were the result of experimentation during the system and architecture development process. Building an architecture involving artificial intelligence with real-world actions involves many steps that could have been improved. Using the architecture, we can see that the model generates real recommendations for users, but the interaction of this user matters in the learning and, consequently, in the use of the system. Thus, the system's training, construction, experimentation, and use were done in a way that reproduced the behavior of users interacting with smart devices in the home as faithfully as possible. The architecture and the lessons learned in this work consider a real use of the system in everyday life with a significant generation of data and actual use of the architecture.

8 Conclusion

Improving smart home services today reflects actions that will take place in the future. How the system may behave in users' daily lives becomes a concern of developers and influences decision-making about a good architecture that will deliver to the user more than home automation but intelligent services for managing the general well-being of residents. The smart home has been evolving through the years, and the user action on systems controlling crucial environmental elements has taken place in smart homes ecosystems that decide what can be better for the residents [7]. Moreover, to keep up with this paradigm shift in automation for increasingly purposeful homes without user actions thanks to artificial intelligence, the evolution in the architecture and implementation of solutions must increasingly consider the support of artificial intelligence and user interactions in everyday life.

Challenges and lessons learned are also crucial for disseminating knowledge about challenges, learnings, and problems and how to overcome them to develop intelligent systems that support the user's routine. With this vision, our study presents an smart architecture and software solution that recommends actions to the user based on real time usage and integrates into smartphones and smartwatches where, with the palm or wrist, they can interact directly with the home. Actions and reactions when building our smart home architecture led us to learn from mistakes and successes that we share in the lessons learned from this work as an invitation for researchers and enthusiasts to increasingly contribute to the state of the art of solutions involving smart homes and devices.

As future work, we intend to evolve the architecture and create redundancies in our cloud services, in addition to understanding some delays in the communication network with third-party cloud services.

Acknowledgements

This material is partially based upon work supported by INES (www.ines.org.br), CNPq grant 465614/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CAPES grant 88887.136410/2017-00, and FACEPE grants APQ-0399-1.03/17 and PRONEX APQ/0388-1.03/14; and FAPESB INCITE PIE0002/2022.

References

[1] Kevin Bouchard, Bruno Bouchard, and Abdenour Bouzouane. 2012. Guidelines to efficient smart home design for rapid AI prototyping: a case study. In *proceedings of the 5th international conference on pervasive technologies related to assistive environments*. 1–8.

[2] Padma Nyoman Crisnapati, I Nyoman Kusuma Wardana, and I Komang Agus Ady Aryanto. 2016. Rudas: Energy and sensor devices management system in home automation. In *2016 IEEE region 10 symposium (TENSYMP)*. IEEE, 184–187.

[3] Xiao Guo, Zhenjiang Shen, Yajing Zhang, and Teng Wu. 2019. Review on the application of artificial intelligence in smart homes. *Smart Cities* 2, 3 (2019), 402–420.

[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[5] J Jithish and Sriram Sankaran. 2017. A Hybrid Adaptive Rule based System for Smart Home Energy Prediction.. In *DIAS/EDUDM@ ISEC*.

[6] Farzeem D Jivani, Manohar Malvankar, and Radha Shankarmani. 2018. A voice controlled smart home solution with a centralized management framework implemented using ai and nlp. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*. IEEE, 1–5.

[7] Stephen Makonin, Lyn Bartram, and Fred Popowich. 2012. A smarter smart home: Case studies of ambient intelligence. *IEEE pervasive computing* 12, 1 (2012), 58–66.

[8] Sunil Manandhar, Kevin Moran, Kaushal Kafle, Ruhao Tang, Denys Poshyvanyk, and Adwait Nadkarni. 2020. Towards a natural perspective of smart homes for practical security and safety analyses. In *2020 ieee symposium on security and privacy (sp)*. IEEE, 482–499.

[9] Prianka Mandal, Sunil Manandhar, Kaushal Kafle, Kevin Moran, Denys Poshyvanyk, and Adwait Nadkarni. 2023. Helion: Enabling Natural Testing of Smart Homes. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2147–2151.

[10] Bauyrzhan Ospan, Nawaz Khan, Juan Augusto, Mario Quinde, and Kenzhegali Nurgaliyev. 2018. Context aware virtual assistant with case-based conflict resolution in multi-user smart home environment. In *2018 international conference on computing and network communications (coconet)*. IEEE, 36–44.

[11] Debayan Paul, Tanmay Chakraborty, Soumya Kanti Datta, and Debolina Paul. 2018. IoT and machine learning based prediction of smart building indoor temperature. In *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*. IEEE, 1–6.

[12] Samad Sepasgozar, Reyhaneh Karimi, Leila Farahzadi, Farimah Moezzi, Sara Shirowzhan, Sanee M. Ebrahimzadeh, Felix Hui, and Lu Aye. 2020. A systematic content review of artificial intelligence and the internet of things applications in smart home. *Applied Sciences* 10, 9 (2020), 3074.

[13] Amit Kumar Sikder, Leonardo Babun, and A Selcuk Uluagac. 2021. Aegis+ a context-aware platform-independent security framework for smart home systems. *Digital Threats: Research and Practice* 2, 1 (2021), 1–33.

[14] Huaizhou Su, Yande Li, and Li Liu. 2018. Gesture recognition based on accelerometer and gyroscope and its application in medical and smart homes. In *Web and Big Data: APWeb-WAIM 2018 International Workshops: MWDA, BAH, KGMA, DMMOOC, DS, Macau, China, July 23–25, 2018, Revised Selected Papers 2*. Springer, 90–100.