MALMÖ HÖGSKOLA

School of Technology

Department of Computer Science

Master Thesis Project 30p, Spring 2015

Design and evaluation of a visual rapid-prototyping environment in an existing smart-home platform

By

Skander Hamada

Supervisors:

Bo Peterson and Daniel Spikol

Examiner:

Romina Spalazzese

Contact information

Author:

Skander Hamada

E-mail: skander.hamada@gmail.com

Supervisors:

Daniel Spikol

E-mail: daniel.spikol@mah.se

Malmö University, Department of Media Technology and Product Development.

Bo Peterson

E-mail: bo.peterson@mah.se

Malmö University, Department of Media Technology and Product Development.

Examiner:

Romina Spalazzese

E-mail: romina.spalazzese@mah.se

Malmö University, Department of Computer Science.

Abstract

Recent advances in the field of Internet of Things (IoT) are placing its own vision, as a platform of interconnected services and devices, at the heart of the smart home concept. This consolidation promises a new wave of innovative products designed in the open, and in which the user takes center stage starting from the very first steps. Therefore, researchers as well as product designers in these increasingly related fields are now tasked with a more complex mission when investigating user behavior. In this thesis we consider rapid prototyping as the upcoming standard process for investigating user interactions in the future smart home. Although past research contributed with several self-contained solutions (built from scratch) to allow such investigations, no accounts were found tackling the problem from our perspective, in which the focus is on how to enable rapid prototyping in an existing proprietary smart home platform by using open standards, software and hardware. To answer this question, we conducted our research with participation of academic researchers and professional designers in the context of an academic and industrial partnership, in an ongoing smart home research project. We used an approach based on the design science research process in combination with the user centered design (UCD) and agile software development methodologies. During this thesis we performed an end to end design process starting from ideation to implementation and evaluation; an architectural blueprint was proposed and a working prototype of our visual smart home rapid prototyping environment (SHRPE) was implemented and tested. The obtained results demonstrate the feasibility of enabling visual rapid prototyping capabilities in an existing smart home platform, by using the system integration process to introduce available open standards, software and hardware tools into the platform. In addition, evaluation results of user testing confirmed that using UCD to iteratively capture user needs in such complex context is a solid approach.

Table of contents

List of Figures

List of Tables

List of acronyms

DSRP	Design Science Research Process
UCD	User Centered Design
SUS	Software Usability Scale
IoT	Internet of Things
WoT	Web of Things
IDE	Integrated Development Environment
GPIO	General Purpose Input/Output
CoAP	Constrained Application Protocol
oBIX	Open Building Information Xchange
POC	Proof of Cpncept
OSGi	Open Services Gateway initiative
DIY	Do it Yourself
WS	Web Services
UX	User Experience
API	Application Programming Interface
M2M	Machine to Machine
WSBPEL	Web Services Business Process Execution Language
OWL	Web Ontology Language
DSL	Domain Specific Language
IoTaP	Internet of Things and People
WAN	Wide Area Network
SHRPE	Smart Home Rapid Prototyping Environment

1 Introduction

1.1 Introduction and motivation

With the rising number of connected devices present everywhere in our daily life, at work, at home and on the go, mindsets of people are shifting, and now expecting the whole to provide a higher value than the sum of individual parts (devices) [1].

Although, considerable efforts are being invested in advancing the Internet of Things (IoT) vision shaping it as a web of interconnected devices, and setting the target to break silos and barriers towards richer interactions, the majority of stakeholders (Industry and Academia) are frequently found absorbed in addressing challenges at infrastructure level [1][2]; such situation fails to satisfy expectations of a growing audience whom on the lookout for breakthroughs that will change how they experience professional, private and social daily life. In particular, smart-home users, whom demand is sustained for flexible services capable of delivering value in a ubiquitous manner and beyond physical boundaries of home walls.

Addressing this new class of user needs requires new ways of working, so far, promises to challenge the status quo have been outspoken in various areas of Information and Communication Technology, and yielded to a wide adoption of the user-centered design approach [3][4][5], especially in mobile, web and consumer products. These culminating successes of the approach in adjacent fields, are now paving the way to its adoption by practitioners within the realm of IoT powered services and smart home platforms.

Inspired by Pahl & Carle [6], we focus in this thesis on smart home applications, and believe that equipping practitioners in this area, with a visual integrated environment for rapidly prototyping new concepts, will cast a positive impact on the design process and its outcomes (products and services). In addition, resulting process enhancements will potentially contribute to breaking down existing barriers towards open and community driven innovation.

1.2 Context

This master's thesis was written in the context of an ongoing research project, "Interaction in the Smart Home: A prototype-driven approach"[1], which is part of the research activities in the Internet of Things and People[2] (IoTaP) research lab at Malmö University. This thesis involved participants from academia as well as a number of industrial partners to IoTaP.

In this thesis, we investigated technical feasibility to enable rapid prototyping of interaction scenarios in the smart home on top of an existing commercial platform. The investigation was limited to cater for the needs of a specific target group of users: academic researchers in IoTaP and professional smart home application designers at industrial partners of IoTaP.

This thesis was performed in a shared context, timeframe and environment, with another master's thesis [7] conducted by my university colleague Emma Mainza. The two theses contributed in the delivery of a proof of concept project hosted by Verisure Innovation AB[3] (Verisure), an industrial partner of IoTaP.

Verisure provided the project with necessary logistics and access to in-house experts in order to support research activities of the two theses in using its commercial smart-home platform (Verisure IQ) as a real world use case.

Joint efforts in the two theses covered planning and execution of workshops involving the same group of research participants. Throughout two of the workshops, data was jointly collected then separately analyzed before sharing and discussing intermediate results. Such close collaboration caused mutual influence between the two theses, and was materialized by the occurrence of several joint decisions, as well as the selection of the same research methodologies. However the two theses proceeded with distinct research objectives and therefore employed the selected research methodologies differently.

[1] http://iotap.mah.se/ishpa/

[2] www.mah.se/Forskning/Sok-pagaende-forskning/The-Internet-of-Things-and-People-Research-/

[3] http://www.verisure.com/

To enforce differentiation in terms of goals, the followed guidelines specified that my thesis is mainly reflecting a system centric view of the problem, and shall focus on backend components to investigate how to design a working prototype based on the system integration approach. The working prototype shall integrate to Verisure IQ platform and expose a minimalistic set of UI components. For the other thesis, followed guidelines specify the subject of the investigation as being the visual programming for the smart home from a user perspective (as opposed to system). However, these clear borderlines did not prevent certain overlaps in results related to the aspects of user research, requirements specification, off-the-shelf software tools selection and design of user interface (UI) components.

The figure below outlines differences between the two theses in relation to their respective contributions to the body of knowledge. My thesis contributed with a proposal for reference architecture and an end-to-end design process which is potentially reusable in future projects within similar context (enabling rapid prototyping for research and open innovation purposes, on top of existing commercial smart-home platform, through integration with off-the-shelf tools). Hence, addressing an identified gap in the literature where the lack of such process is observed.

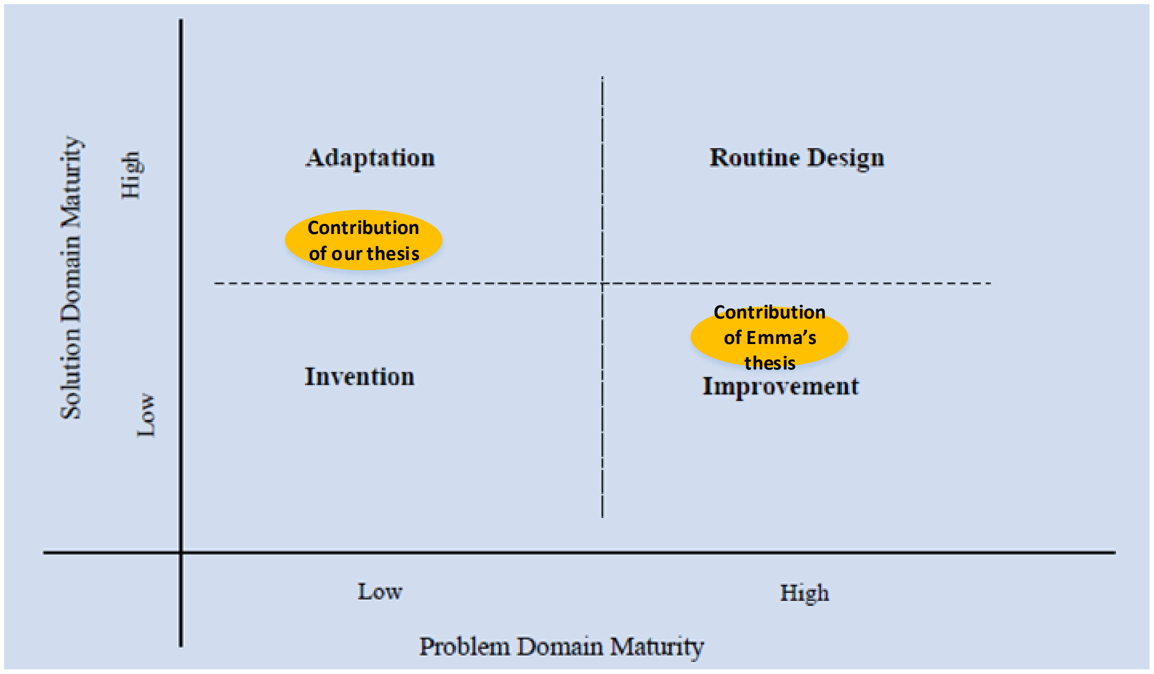


Figure 1: Contribution differences of the two theses (Adapted from [8]**)**

1.3 Aims and objectives

The aim of this thesis project is to investigate feasibility to build a visual rapid prototyping environment on top of an existing smart home platform. Our investigation addresses mainly the technical feasibility aspect. Hence, in order to generate meaningful findings, we restricted the selection of technical components to using and adapting state-of-the-art tools and protocols in the area of IoT.

Objectives of our work are broken down as follows:

- Investigation of needs of the target group (mentioned above) through workshop based user-research.
- Analysis and specification of software requirements.
- Review of existing graphical prototyping tools.
- Investigation of basic interaction patterns between connected devices in a smart home environment.
- Implementation of a proof of concept for a visual rapid-prototyping environment.

- Evaluation of the implemented proof of concept through conducting scenario-based tests to assess conformity against functional requirements.

1.4 Research questions and hypotheses

Aims and objectives of this theses, as stated above, are multifaceted and hence require a host of different research activities. To accommodate this, our research questions are designed to allow certain flexibility to combine exploratory and explanatory inquiry approaches. This is achieved by a two track strategy, where the first track investigates technical feasibility, and the second track inquires methodological adequacy.

Below we present our two research questions, classified as main question and sub-question. Each research question is relevant to one of the two tracks, and associates to it a working hypothesis.

- **Main research question**: How can we enable visual programming and execution of cross-device interactions for rapid prototyping purposes in an existing smart-home platform?
    - **Hypothesis**: Adapting and integrating an existing graphical wiring tool, is a possible solution to enable rapid prototyping of home automation scenarios in an existing smart-home platform.
- **Sub-question**: How can we combine agile software development with the user centered design approach to guide the design process of a rapid prototyping platform for the smart-home?
    - **Hypothesis**: the user centered design approach can be used to iteratively investigate user needs and functional requirements, as part of the design process towards a visual prototyping environment for the smart-home.

1.5 Limitations

This thesis was conducted at the intersection of two ongoing umbrella projects (academic and industrial). The nature of this context conveyed strong anchoring in real world situations, and benefited validity of outcomes. However, this setup resulted in increased operational complexity, considering that the two stakeholder organizations are operating under different working processes, expectations, and constraints. As a consequence, it required extra efforts in order to decide on process related tradeoffs.

In the following we summarize the main points:

- Agile way of working with the industrial stakeholder (acting as product owner), the scope of the project itself was subject to continuous changes and adjustments.
- Software architecture of the proof-of-concept should be designed as a non-intrusive layer to the existing smart home platform (VerisureIQ).
- Outcomes should include a proposal of Middleware architecture foundation to support future integration with the VerisureIQ platform.
- The user centered design approach shall be leveraged in the research process to fit into the strategy of the umbrella academic project.
- Limited number and availability of research participants throughout the project, which leads to conservative generalization of outcomes.

1.6 Ethical considerations

Involvement of human subjects in conducting research is historically a common practice in the fields of medicine and social sciences, justified by its central role as subject in the underlying scientific investigations. By the time, evolution of societies and the rise of human rights awareness have led to the definition and legal enforcement of a host of ethical codes of conduct targeting research activities in these two fields [9]. Despite the existence of numerous variations in research codes of conduct, the Nuremberg code [10], originally devised for medical experiments, so far represent the most influential proposal; its fundamental concepts (voluntary consent, the priority is for the security and wellbeing of subject, liberty of the subject to withdraw from the research at any time) can be traced in all modern ethics guidelines of research. For instance, the Helsinki declaration of World Medical Association [11] builds upon the Nuremberg code and includes articles related to the broader definition of human rights including privacy, confidentiality and protection of vulnerable social groups.

In 2006, Wright [12] highlighted a set of major present and probable future ethical issues in the field of computer science and engineering, and identified human subject participation as an important source of such issues. But unlike in medicine and social sciences, participant-related ethics in computer science and more generally engineering research are rarely regulated [9] [13]. A notable exception consists in research funded by the European Union (EU) for which it enforces compliance to “EU human rights charter and convention” [14] [15]. Such new regulations in addition to the immerging trend of user centered research in computer science [16], are now acting as a motivator for the research community to seriously consider ethical issues in computer science and engineering research [13] [9].

In our thesis we considered ethical issues in connection to the involvement of human participants, and took necessary measures to guarantee the rights of participants. For this, we undertook following actions:

- Verbal voluntary consent was acquired from each participant.
- Participants were acknowledged that participation is not binding and therefore they can withdraw from the research activities at any time.
- Participants were informed whenever photos are taken and voice being recorded.
- All collected information was anonymized.
- Only necessary information to the research was collected.
- Collected information (photos and voice recordings) were saved in a password protected medium and access was restricted to authorized persons only.

1.7 Thesis outline

The rest of this thesis is structured into four parts. In the first part we present the background from related areas to our research (chapter 2), and define the research methodology used to conduct the work (chapter 3). The second part presents intermediate results from the two research aspects: user research (chapter 4) and software engineering (chapter 5). The third part presents the final evaluation results (chapter 6) of the two research aspects mentioned previously. Finally, in chapter 7 we state the conclusion and present gaps for potential future work.

2 Background

Expanding on our research questions stated above, we acknowledge the emerging structural relationship between smart-home technologies and IoT. In this context of rapid prototyping for the smart-home, IoT is expected to fulfill the connectivity enabler role, towards an open platform on top of which we can program and run smart-home applications. As a concept, IoT is still in its early stages, despite the emergence of a handful of protocols in the field, targeting mainly industrial and smart-buildings applications, there has not been yet one prevalent standard which reached the milestone of industry wide adoption. Therefore, heterogeneity in terms of hardware, software application layer and communication protocols, still characterizes the IoT ecosystem.
In the following, we synthesize the information collected during our investigation of state-of-the-art research in the areas of smart-home, IoT, as well as visual programming of physical device interactions. This focus aligns with our aim to deconstruct barriers towards the adoption, of prototype-based research practices, by smart-home interaction researchers and concept designers.

2.1 Smart home architecture

In our review of the literature, we found that most of modern home automation systems define four types of entities. According to Mazhelis and Tyrv [17], these entities can be mapped to corresponding separate physical devices or combined together depending on the system’s design. For instance, we consider the following abstractions: (1) sensor or actuator device, having “short range” wired or wireless connectivity and low computing power. (2) gateway device playing the roles of device/service registry, communication enabler between local devices and a secured proxy to external systems or entities via internet. (3) web or mobile user interface for interacting with and configuring the system. (4) service component for executing application logic, usually hosted in a cloud platform.

Following the same direction, Pahl and Carle [6] elaborate on a generalized conception, advancing that architecture of home automation systems can be abstracted using three logical layers. (1) Hardware layer comprising sensors and actuators, (2) middleware layer and (3) service layer. In the rest of this report, we adopt this classification and place our focus on the service layer.

But before moving forward, we attribute attention to the work done by Dixon et al. [18] consisting in a third approach to smart home architecture, leading to discussion around challenges faced by current users of smart-home systems. In this respect, Dixon et al. [18] identified three main challenges, the first being related to management, where they argued that system configuration features primarily intended to casual users are actually being inappropriately designed, requiring from the user, extensive functional - sometimes technical - knowledge of the system. A second challenge stemming from the desire of users to accommodate emerging needs when defining applications in real life setting. And the third is roadblocks encountered when extending the system by adding new - potentially heterogeneous - devices. As an attempt to address those challenges, Dixon et al. [18] devised a framework for abstracting such system components; they proposed a “PC like abstraction” for smart home middleware with three layers (management, device functionality, and device connectivity) in addition to a fourth service (or application) layer. This choice was motivated by the fact that PC operating system abstractions have been successful in dealing with similar challenges in a comparable technical and organizational environment. We are referring here to an environment characterized by heterogeneity of components on the hardware level, and a software ecosystem dominated by third party providers.

2.2 Applications for the smart home

In this thesis, the focus is put on the application layer in smart-home platforms. To investigate the different approaches, we use a conceptual model inspired by the work of Jung et al. [19], in which the authors synthesized different existing approaches for implementing control logic in IoT infrastructures. According to our model, we classify application layer design based on two attributes, the first represents the expression method of control logic (script based, graphical based, or hybrid) through which designers or end users could define applications. The second represents the deployment and execution strategy (centralized or distributed), which defines roles of each physical and logical node in the system. In relation to this, and from a semantic point of view, we observed, while reviewing relevant literature, a disparity in the terms used by different authors in reference to the act of defining applications for the smart-home. Although these terms are used interchangeably in certain contexts, they are in fact embedding implicit references to the adopted architecture paradigm. We identified for instance the following variations, “service orchestration”, “service mashup”, “automation task”, “application” and “control logic”.

2.3 Execution platforms

Kao and Yuan [20] argue that until recent times, the Open Services Gateway Initiative (OSGi) architecture represented the prevalent service oriented architecture in smart-home systems. OSGi offers the foundation, on top of which componentized platforms can be built. According to the standard specification, actual implementations of OSGi should provide the following features: container managed service life-cycle and discovery, and dynamic (runtime) deployment capabilities of service packages, called “bundles”. Despite merits of such capabilities, OSGi failed to define standard semantics for control logic, and exhibited limitations in use cases requiring exposure of deployed services to heterogeneous external entities.

Efforts to circumvent such limitations have led to IoT technologies being underway in shaping the new infrastructure for the smart-home. Furthermore, advances in IoT towards the Web Of Things (WOT) vision placed the web, in particular REST based services, at the heart of modern platforms. Supporters of the WoT vision are now driving ongoing initiatives to standardize existing IoT protocol stacks, and materialize industry-wide interoperability of connected devices. However, Kovatsch, Lanter and Duquennoy [21] demonstrated that outcomes of these efforts are still fragmented; convergence is mostly impeded by the fact that majority of marketed devices fall under the “mote class”, a class regrouping devices with constrained resources, implying inability to accommodate required energy and computing power for exposing REST interfaces. In most cases, mote-class devices need to rely on gateways to handle REST requests on their behalf, then assure tunneling over a specific protocol with smaller foot-print. For instance, WoTKit [22] illustrates such implementation scenario.

In spite of these initial limitations, recent advances in the WoT realm have led to the development of a new protocol: Constrained Application Protocol (CoAP). CoAP is a light-weight version of HTTP-based REST architecture, primarily designed for devices with constrained resources, and guarantees simple binding to plain HTTP. Based on CoAP, both Actinium [21], and IoTSyS [19] materialize an important milestone of the WoT proof of concepts. Through IoTSyS, Jung et al. [19] argued for an architecture which enables distributed execution of control logic, and offers facilities to integrate with legacy systems. The proposed solution supports graphical expression of control logic, and is based on existing standard protocols, namely, IPv6, oBIX, and CoAP,

2.4 Control logic representations

In this section, we reviewed various representation techniques of control logic, which were found in past research and commercial products. The classification of each reviewed instance is based on two conceptual aspects: the underlying programming paradigm and the supported input method. We derived these aspects from the notion of "expression methods" as defined in [19].

The focus of this review is on the facilities, offered by smart-home systems, enabling designers to rapidly prototype applications and service mashups.

Blackstock and Lea [22] used the data flow paradigm in building the WoTKit, where functional pipelines are built by combining elementary modules, each module implements a basic processing logic that produces output based on input data. As user input method, a graphical programming environment is included, users can express desired control logic using the graph metaphor.

Chin, and Winckles [23] applied the rule based paradigm in conjunction with the Pervasive interactive Programming input method in a home automation setting. End users are able to customize the behavior of connected appliances in their environment by embodying the scenarios physically, and manipulating objects so that the system learns their intentions. The system captures events and translates the context internally into a collection of rules.

The same rule based paradigm, is commonly used across a variety of IoT platforms, another illustration is the cloud hosted automation engine IFTTT [24]; its UI enables users to define automation rules by composing 3 types of elements: triggers, events and actions. In contrast, according to Rietzler et al. [25], this approach potentially leads to unmaintainable and conflicting repository of rules, hence undermining stability and predictability of the system.

From a different perspective, Rietzler et al. [25] propose a process-driven approach to programming smart homes. With the help of a graphical interface, users can express complex automation processes by means of graph representations, every process flow can involve a multitude of events, devices, actions and timing elements. Unlike other approaches, here, contextual cues are implicitly inferred through analyzing defined transitions between the process steps, which results in a higher immunity against inner state conflicts. In their work on USHAS, Kao and Yuan [20] adopted the same process-driven approach, in particular the standard implementation for web services: Web Services Business Process Execution Language (WSBPEL). USHAS design exhibits a focus on the semantic aspect in programming home automation processes, and therefore includes an OWL ontology as abstraction of the domain model. Classes belonging to this ontology are used to express automation processes, as well as in representing global state of the system. Examples of high level USHAS classes are: “Person”, “Device”, “Time”, “Environment”, “Event” and “Location”.

In continuation with the same family, we introduce the linear approach, a more traditional approach when compared to its process-driven successor (presented earlier). The linear approach comes in two varieties, visual or text based; the text based variation, known as scripting, allows users to write applications using a variety of languages. While the visual variation, conveys richer user experience through graphical metaphors of conventional script elements. This explains the reason behind the relative wide adoption of Jigsaw editors in programing tools targeting the end-user.

Besides, concrete examples of text based scripting were materialized in [21] via selecting JavaScript as the scripting language for their platform, and in [19] where Python was used to give users the possibility to define reusable processing components inside applications. Although, both alternatives (textual and visual) are efficient in expressing linear control logic, they present comparable limitations in handling context, timing and concurrency [25].

Lastly, we shed the light on the model driven approach, for which Sánchez et al. [26] argue for a framework that covers the full lifecycle of home automation projects. A model driven development approach was utilized to (1) express and catalogue generic requirements with the help of a graphical Domain Specific Language (DSL), (2) execute model transformations, and (3) generate deployable code. Additionally, an emphasis on software engineering best practices (reuse, maintainability, traceability, proper tooling) was demonstrated.

2.5 Evaluation methods

Throughout the reviewed papers, we encountered an array of methods used to evaluate smart-home projects from a rapid prototyping perspective. They vary on a large spectrum where the focus is put at one end, on system performance metrics, and at the other end, on usability.

To evaluate their work, Chin, Callaghan and Winckles [23] used two different methods, experiment and survey. The series of experiments involved 18 participants, each participant was asked to customize a given smart-home system using the physical programming approach, then respond to a questionnaire, therefore provide data needed to assess usability of the approach. The second evaluation technique, an online survey, was conducted on a larger scale, 69 participants responded to the questions gauging their perception of various end-user programming approaches, for instance, script based, graphical based and tangible (physical based).

Surveys can also be used to evaluate other aspects related to the smart-home, such as software design frameworks and methodologies. For instance, Sánchez et al. [26] examined, through software engineering lenses, the common issues in designing smart-home platforms. Then devised a framework promoting model reusability, and evaluated outcomes using a survey, for which the target group was home automation developers.

So far demonstrated, challenges to evaluate smart-home related artefacts, are not limited to final products, Edwards et al. [27] argue that in reality it expands to “infrastructure software”, in reference to software development toolkits and environments. Therefore the need for elaborate evaluation approach. To devise such evaluation approach, Edwards et al. [27] started from the observation that features and style of end-user applications are directly influenced by facilities offered in underlying development toolkits. The proposed approach is materialized in a set of guidelines adhering to principles of user-centered design. It dictates a process where the toolkit under evaluation is employed to build several simple end-user application prototypes. After that, the evaluator perform individual evaluations of these prototypes, and finally combines outcomes of these sub-evaluations into one synthesized macro evaluation, which is subsequently used as a proxy to perform usability and feature-set assessment of the toolkit. In the case where objectives of the overall evaluation are not limited to the user-centered approach, the evaluator should complement this strategy with another one focused on system performance. Kovatsch, Lanter and Duquennoy [21] used such evaluation method to evaluate their open source prototype from a computational performance point of view.

2.6 Conclusion

Past research addressed software and network related problematics across various layers of smart-home platforms. These contributions not only covered the backend, but also addressed programmable front-end interfaces which enabled rapid prototyping or end-user configuration. However, previous contributions focused on designing standalone systems and implemented software from the scratch; such approach does not accommodate technical and time constraints commonly encountered in projects at the confluence of research and industry. Considering the case of projects aiming to introduce rapid-prototyping based research as innovation catalyst within smart-home commercial platforms, we acknowledge that current literature does not cover necessary methodological frameworks, neither system integration guidelines, which are capable of guiding an implementation of end-to-end rapid prototyping capabilities.

In this thesis we conducted an empirical investigation around these aspects, and demonstrated the feasibility of transforming a closed platform (walled garden) into an open platform which supports rapid-prototyping activities. Therefore, the current report serves the purpose of communicating around our process to design such solution. The process establishes a foundation for future reuse and adaptation to similar contexts, hence, encouraging other smart home solution providers to adopt a more open approach.

3 Research methodology

In the light of our research questions, we acknowledge two main aspects of the overarching research problem; 1) technical aspect, relevant to software engineering [28], where the objective is to demonstrate feasibility by building a working software artefact; 2) methodological aspect related to investigating adequacy of employing a user centered design approach (UCD) [29][30][31] to support the design process of the software artefact. To cater for these two aspects, we selected the Design Science Research Process (DSRP) [32][33][34] as the overarching research process of this thesis.

DSRP is an established research methodology in Computer Science, it suggests a sequential process comprising six steps which must be carried out in a relevant order according to the research goal. Flexibility to choose the starting point is enabled by the definition of four possible entry points to the research process, each labelled with a generic research aim.

In this thesis, the two research aspects, 1) technical and 2) methodological, are mapped to "design and development centered approach" - one of the four entry points defined in DSRP. The ability to perform this direct mapping, demonstrates fittedness of DSRP to our research problem.

However, addressing other aspects related to the project's context, requires extending our research methodology beyond DSRP. We hence adopt a sequential mixed (quantitative and qualitative) strategy as described in [35]. The qualitative path is mainly used to address our research sub-question, helping in requirements analysis and elicitation through a workshop based approach, and in identifying nominal scenarios which our developed artefact should adhere to. These scenarios are then used as a benchmark for measuring compliance of the built artefact and represent an anchor point towards the quantitative path. From a qualitative perspective, data collection will mainly take place during two workshops with the help of the following methods: paper prototyping, semi-structured interviews and observation.

As for the quantitative path, it will serve in answering the main research question, thus measuring functional and non-functional compliance of the developed artefact. This is done through evaluation activities, where candidate methods for data collection are: questionnaires and scenario-based testing.

Lastly, we see the need for adding a software development methodology layer to our process, as the project involves real world stakeholders. “Internet Of Things and People” researchers represent the main target group, and Verisure accomplishes the role of the product owner. Two main concerns should be addressed; first, unclear and changing requirements; second, undocumented technology elements from the product owner are going to be used for building the artefact. Taking into consideration these concerns, has led to choosing the agile methodology [28]. Agile principles encourage iterative work and delivery, and enforce a deep involvement of the stakeholders – notably product owners – throughout the project lifecycle; therefore enabling efficient steering of the project’s and control over its scope.

3.1 Three phased research process

In the following, we summarize the overall research process by means of three phases; for clarity considerations, we use the term “designers” in reference to academic practitioners and concept designers.

1- Workshops with designers to discuss envisioned home automation scenarios, current and future workflow for developing new concepts; qualitative data shall be collected and benchmark scenarios identified then refined.

2- Concept development and Implementation of the prototyping environment following the agile methodology. Development of the artefact shall satisfy emerging requirements, which result from the ongoing analysis of a) data generated by participants (including step 1), and b) technical constraints of VerisureIQ platform.

3- Functional evaluation of the implemented prototyping environment with the involvement of participants. In this evaluation, we will use the methods: questionnaire and scenario-based testing.

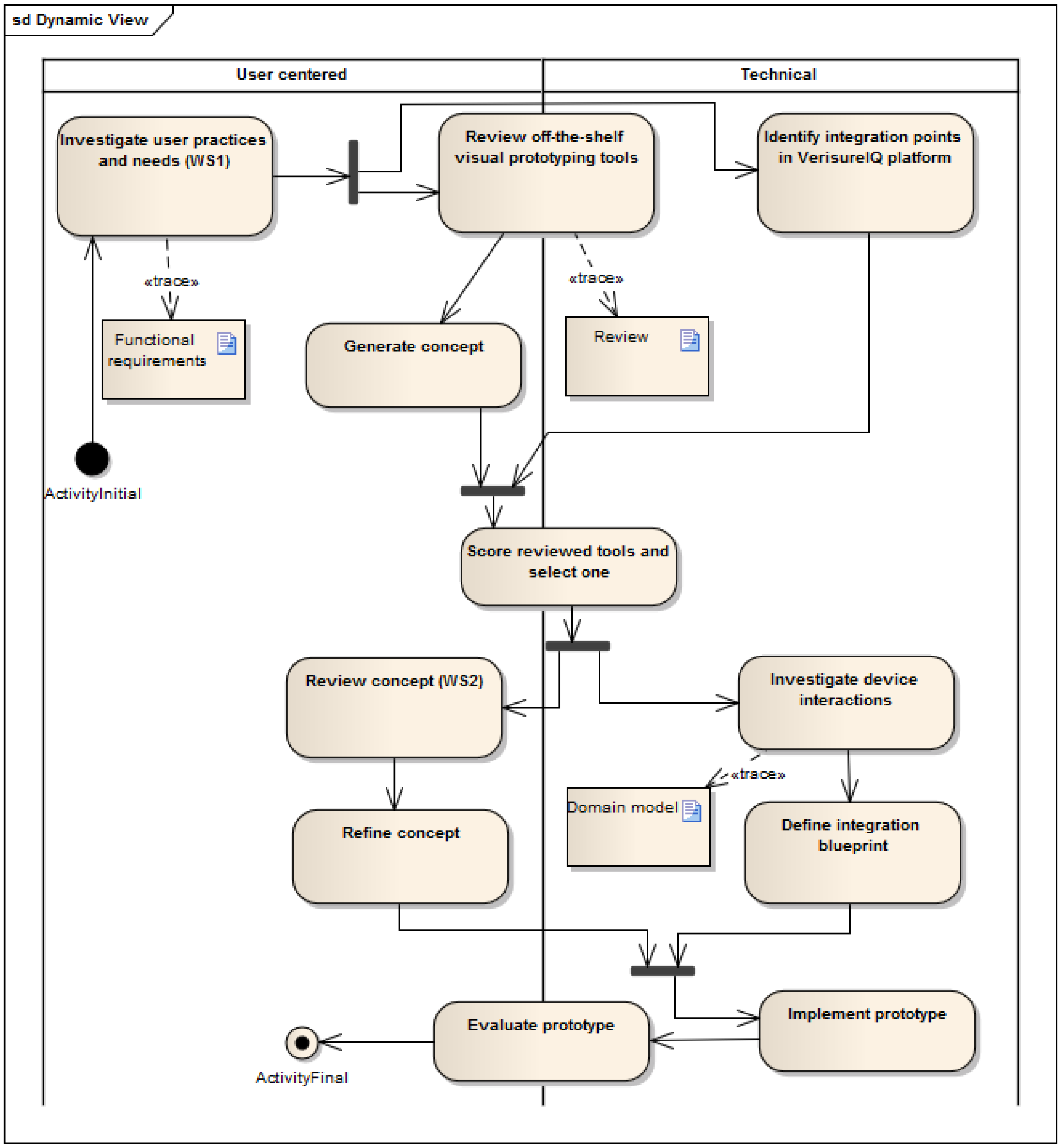

sd Dynamic View
User centered
Technical
Investigate user practices and needs (WS1)
Review off-the-shelf visual prototyping tools
Identify integration points in VerisureIQ platform
«trace»
Functional requirements
Generate concept
«trace»
Review
ActivityInitial
Score reviewed tools and select one
Review concept (WS2)
Investigate device interactions
«trace»
Domain model
Refine concept
Define integration blueprint
Evaluate prototype
Implement prototype
ActivityFinal

Figure 2: Research process activities

3.2 User research through UCD

Guided by our research process, we will use the UCD approach to tackle the first and partially the second phase of the process, as summarized in the previous section. Through this setup, we establish a direct link to our research sub-question and its hypothesis, for which we will perform our investigation by designing and facilitating a series of three workshops. The first two are for gathering initial user needs and refining functional requirements, and the third is for evaluating the combined agile and UCD design process by proxy of tangible outcomes (software artefact). In addition, since validity of such evaluation is dependent on whether participants satisfy representativity of the intended target group, as stated in (section 1.2 Context), we specified that participation number should be comprised between six and eight, and limited the target profile of participants to a list of relevant backgrounds as follows:

- Researcher within the Smart-Home project at IoTaP research center.
- Interaction designer or related.
- Service designer.
- Product specialist in the area of smart home.
- Concept or application designer in the area of smart home.
- Smart home hobbyist.

In the following sub-sections, we present aims and research methods, of the first two workshops.

3.2.1 Idea generation workshop

<u>Workshop aim</u>

For this first workshop our aims are to:

- Invite participants to collaboratively generate a set of scenarios, those scenarios should represent concrete examples of smart home applications, in which they are interested to prototype further.
- Probe current prototyping practices which participants are currently using, when they are developing and communicating new concepts for the smart home.
- Investigate participants about their intended use cases of a visual rapid-prototyping environment for the smart home.

We will perform these investigations at a high level of abstraction in order to prevent influencing participants with our assumptions regarding appearance and functioning of such visual prototyping environment. Nonetheless, we will introduce in this workshop, examples of elements which belong to the domain model, to help frame the discussion and better engage participants.

Data collection

Data collection will be performed through a semi-structured interview, and a paper prototyping activity in which participants will visually express scenarios in the smart-home.

3.2.2 Concept validation workshop

Workshop aim

In this second workshop the aim is to introduce our initial proposal of the visual prototyping tool which will result from the first design iteration, and investigate the expression power it provides to users. This artefact should materialize a customized version of the visual programming tool which will be selected during the project and described later in section 3.3.2.

Compared to the first, this second workshop will be conducted at a lower level of abstraction and based on the selected prototyping tool (section 3.3.2), with emphasis on details and high fidelity representation (wireframes) of actual graphical elements and execution logic.

Data collection

Data collection will be performed through a paper prototyping exercise, in which participants will visually prototype a set of predefined smart-home scenarios using a set of building blocks. These building block should represent extensions to the visual prototyping tool (section 3.3.2) and will be included or excluded accordingly to support testing different assumptions.

Throughout the exercise, we will register success and failure of participants while they attempt to visually prototype each given scenario using the provided building blocks. In addition, information related to potential new building blocks created by participants, to achieve their goal in the exercise, should also be recorded.

3.3 Design and development centered approach to DSRP

3.3.1 System integration approach

As stated previously in this chapter, DSRP constitutes the foundation of our research methodology, where the "design and development centered" [32] approach will be used to tackle the third and partially the second phases of the overall process (see section 3.1). In practice we will adopt a system integration approach to test the hypothesis of our main research question (see section 1.4), therefore we define the following steps which should be performed in a sequential manner:

- Reverse engineering of existing Verisure platform with the adequate level of abstraction.
- Identification of technical constraints.
- Identification of possible integration points.
- Review and selection of an off-the-shelf visual prototyping tool.
- Elaboration of the integration architecture blue-print.
- Design and implementation of a functioning prototype (following the Agile principles).
- Evaluation of the prototype.

3.3.2 Multi-criteria decision making method

In software engineering, a common problem encounters practitioners when there is need to review a number of candidate software components and select one to become part of the system under design. In this thesis, we will use a Multi-Criteria Decision Making (MCDM) called the Hybrid Assessment Method (HAM) [36]. All MCDM methods materialize a formal process and generate results based on a combination of the practitioner's technical judgment, and the computational algorithms specific to each

method. HAM method describes a process of 5 steps and defines an end-user scale with discrete values in [-8, -6, -4, -2, 0, 2, 4, 6, 8].
In the following sub-sections, we summarize the steps of HAM as they will be applied in reviewing and selecting the visual prototyping tool for SHRPE.

1- Criteria and candidate tools

In this first step of the process, requirements and possible integration points with the existing platform should be analyzed then translated into high level assessment criteria. With the help of these criteria, we will pre-select 12 candidate tools consisting in off-the-shelf prototyping tools, which target the following application domains: home automation, IoT, interactive user experience, education of programming and industrial process control.

2- Criteria prioritization and weight calculation

In this step we will consider each pair of criteria and determine the differential value of their importance according to our understanding of the problem, and in reference to the thesis context and the main research question. These vales will be expressed according to the HAM scale and stored in a matrix format. The matrix will then be processed using an algorithm defined in HAM to calculate the weight of each criterion; higher weights imply bigger importance.

3- Assessment of candidate tools and overall rating

In this last phase, we will use the HAM scale to express the extent a given tool is satisfying each of the criteria. We will develop a set of guidelines upon which such assessment is based. In practice, this will imply iterating through the list of reviewed tools and populate the HAM matrix elements with the corresponding level of each (tool, criteria) pair. Then process the resulting matrix using HAM rating algorithm, which calculates the rating of each candidate tool in percentage figures. A higher percentage indicates a better adequacy to the context.

4 Software requirements

4.1 Introduction

In this chapter we begin with presenting relevant findings from the two workshops "idea generation" and "concept validation" in which respectively 6 and 4 participants have attended and actively contributed according to the agenda of each workshop. Then based on these findings, we elaborate on the list of functional requirements which constitute the initial product backlog of SHRPE, in addition to the list of non-functional requirements.

4.2 Findings from UCD workshops

4.2.1 Current prototyping practices

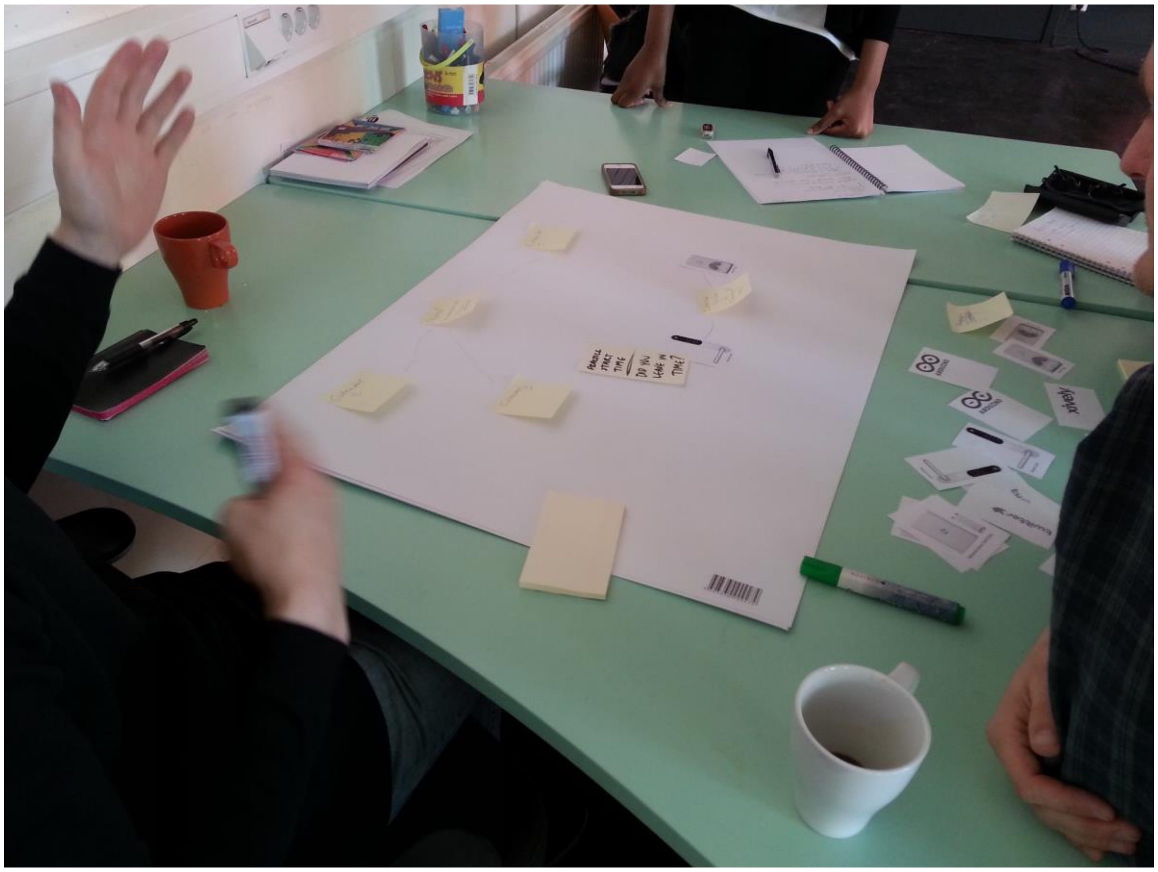


Figure 3: Participants discussing their prototyping practices

Based on responses of research participants regarding the methods they regularly use at work, we identified 8 different methods for concept prototyping and illustration. Further analysis of data resulted in splitting these methods into two categories "static" versus "interactive" depending on the nature of outcomes of each given method. The summarized results can be found in the table below; noting that method, respectively category definitions, represent mutually exclusive classes, and that a given participant's answer can generate a multitude of observations, each belonging to a different class.

In order to eliminate resulting redundancies when mapping method column to category column, for example in the case of a participant using two methods belonging to the same category, we applied a filter rule, which enforces that: at most one occurrence per participant for a given category will be taken into account, even if several have originated from a same participant and targeting the same given category.

Category/Approach	Methods	(Method) Number of occurrences	(Category) Number of occurrences
Static (non-interactive)	Use cases/scenarios/slide show	2	5
	Low-fi prototypes	4	
	UI Hi-Fi prototypes	3	
	Form factor prototypes	2	
Interactive	UI Interactive click through	2	4
	Software+hardware working prototypes	1	
	Hardware prototypes	3	
	Simulation/wizard of Oz	2	

Table 1: Questionnaire used in workshop 1

Collected data also revealed the existence of two distinct approaches to prototyping, the first is an approach which involves programming, where at least one programming or scripting facility is used. The second is an approach not involving any programming skills; both approaches gained equal adoption rate among a sample of 6 participants. Another insight from the same data set revealed that (66%) of participants previously considered user testing techniques as part of their design process.

Following the same principle as in the previous table to report observations, we summarized the results related to usage patterns of tools as reported by the participants.

Category	Tools	(Tool) Number of occurrences	(Category) Number of occurrences
Visual presentation	Power point	2	4
	Adobe illustrator	3	
	Google docs drawing tool	1	
web/mobile UI prototyping	Flinto	1	1
	JustinMind	1	
	Invision	1	
Hardware	Arduino	3	3
	Lego mind storm	1	
	Existing smart-home devices	2	
Software	Programming languages	1	2
	Scripting languages	1	
	Free APIs (IFTTT and similar)	2	

Table 2: Currently used prototyping tools

4.2.2 Four categories of smart home objects

To categorize building blocks of smart home applications, we analyzed the content generated by our research participants in the workshops. The content illustrated envisioned scenarios in the smart home, and revealed four categories into which building blocks (or objects) can be grouped. The following table presents related details.

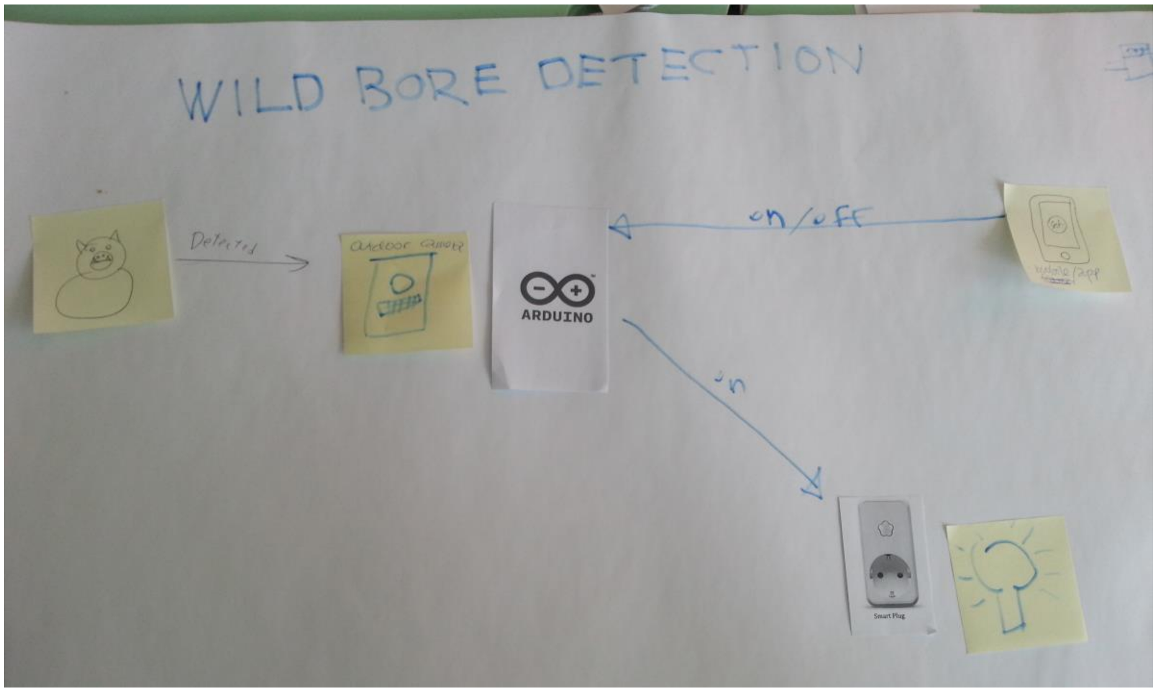


Figure 4: Example of a smart home scenario generated by a group of participants

Scenario name	Verisure stock device	Prototyping device	UI facility	External API
Presence aware heating regulation	-Climate sensor -PIR	-Audio beeper -Heating system controller based on Arduino	-Interface to input desired temperature	-Xively
Sync habitants online calendars with home billboard calendar		-Physical billboard calendar	-Configuration interface	-Google calendar
Teenage coming home late tracking (version 1)	-SmartLock (controlled door lock)	Scent diffuser	Interface to turn ON/OFF	-IFTTT -Twitter
Teenage coming home late tracking (version 2)	-SmartLock (controlled door lock)	-Flick button -Alarm (beeper)	-Interface to set conditional alarm time	
Wild boar presence detection	-SmartPlug with lamp	-Outdoor camera with Arduino	-Interface to arm/disarm	
Elderly activity monitor (analyze activity	-PIR	-Mic (sound level detector) -FLIR	-Configuration interface	-Cloud hosted AI

and trigger alerts)		-Flic button		algorithm

Table 3: Four categories of smart home objects

4.2.3 Expressed user needs in semi-structured interviews

As the following table suggests, needs in terms of features expressed by our research participants can be grouped under 3 categories. We interpret the emergence of these categories as to highlight areas of interest or concern, to which participants attribute high importance. Category 1 embodies the need for an interactive UI capable of assisting the user in a ubiquitous manner, rather than intrusive methods commonly used in visual tools, for instance “automated software wizard assistants” [37]. Putting emphasis on workflow efficiency aspects, Category 2 underlines the need to support collaboration among users, in addition to facilitating reuse of created models. The last Category 3 addresses the core requirements related to expressing smart home automation logic and control. Majority of suggestions belong to basic binary logic and control operators, which prevail amongst programming languages and building blocks of digital electronics. However, one entry makes the exception, it relates to the context-aware approach for programing interactions among domain objects.

Category	Sub category	Feature
<u>Category 1</u> **System accessibility and user assistance**	Built into the UI	- Overview of possible input and output visible at widget level. - Assisted drag and drop (possible links are highlighted while dragging a widget). - Widget appearance mirrors the form factor of corresponding physical device. - Dynamic history which suggests recently used widgets.
	Supported by user community	- Community driven forum, in which users are invited to help each other.
<u>Category 2</u> **Productivity and collaboration**		- Saving and loading scenario templates - Collaboration workflow (traceability of contributions and contributors, managing annotations, version control)
<u>Category 3</u> **Language elements**	N.A.	- Contextual events (example: coming home, leaving home) - Counters - Timers - Binary logic gates (AND, OR, XOR…) - Conditional gates

		- Threshold gates

Table 4: Features demanded by the particiapants

4.3 Initial product backlog

Based on the context of our project, and taking into account findings from the two workshops, we formulated the following functional requirements for the visual rapid-prototyping environment being designed (SHRPE).

FR1: Drag and drop visual representations of smart-home devices

FR2: Drag and drop visual representations of external services

FR3: Drag and drop visual representations of basic control logic

FR4: Auto-populate design interface with configured entities (mentioned in FR1) in the user smart home installation

FR5: Express scenarios through visually linking entities mentioned in FR1, FR2 and FR3

FR6: Deploy and execute scenarios

FR7: Export a scenario after finishing its design

FR8: Import a saved scenario

FR9: Support of a popular scripting language

4.4 Non-functional requirements

NFR1: Isolation of user environments

NFR2: User interface accessible using a web browser without additional dependencies

NFR3: User interface accessible from WAN

5 Artefact design and development

5.1 Introduction

In this chapter, we refer to our main research question to investigate possible methods for adding a visual programming layer to an existing commercial smart-home platform. Although a variety of strategies can be used to achieve such goal, we limited our investigation in the current thesis to the software system integration approach. Our hypothesis materializes this choice and steers our research towards an attempt to prove feasibility of such approach in this specific context. We further restrict the context to, 1) using state-of-the-art visual prototyping tools, 2) designing generic and loosely coupled software architecture to integrate with the legacy platform.

5.2 Reverse engineering of legacy

In this section, we present a simplified model describing the current architecture of VerisureIQ platform.

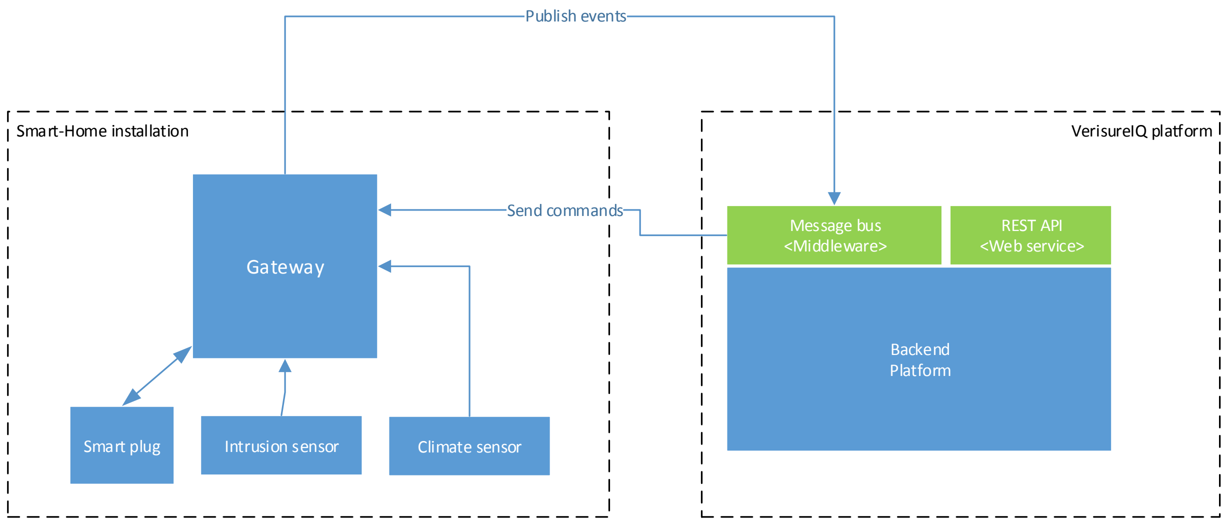


Figure 5: Architecture overview of VerisureIQ platform

The VerisureIQ platform represents the backbone of the existing system, it assures core business services including: user registry and authentication management, provisioning and control of customer installations (gateways, devices), alarm and security oriented services, real-time monitoring, customer billing, and other.

However, in the context of our thesis, we only considered the middleware functions it covers, in particular, functions related to control and event handling of connected devices behind the gateway. The figure above, depicts the structure and communication topology linking a smart home installation to the platform. Connected devices such as SmartPlug or ClimateSensor communicate with the gateway via a 2-way wireless link, used to send events triggered by their environment or receive control commands. In this topology, gateways are responsible to federate and perform specific logic on received data if necessary, then decide on forwarding the flow into the appropriate direction, this can be either towards devices or the platform. In the latter case, an enterprise grade message broker assures publish/subscribe asynchronous communications (based on the AMQP protocol).

5.3 Integration points

Based on outcomes from the reverse engineering step, we identified the following integration points in Verisure platform:

Integration point	Potential	Limitations and considerations
Message bus	Allows message oriented asynchronous communication using the publish/subscribe pattern. It supports message routing function and message payloads are able to reach installed gateways and other backend modules.	Message bus cannot be accessible from the WAN. Message bus resources should be utilized in an optimized manner.
REST API	Offers secure synchronous interactions with the platform. It exposes installation centric services for structure and status querying in addition to a limited number of commands.	In order to communicate with a given installation (Gateway), the user should be authenticated against that specific installation. Target architecture should not result in any changes to the REST API.
Gateway	Offers proximity Radio Frequency (RF) communication, based on connectionless link through which packets are	Packet size is very small. Proprietary protocol.

	transferred. Used to communicate with Verisure stock devices, for example, SmartPlug, ClimateSensor and others.	

Table 5: Integration points in Verisure platform

5.4 Review and selection of the visual prototyping tool

We used the Hybrid Assessment Method (HAM) [36] to assess then select the visual prototyping component of SHRPE. In the following we present intermediate and final results of the selection process as described in (section 3.3.2).

1- Criteria

In this first step of the process, we analyzed the functional requirements, non-functional requirements and integration points of the existing Verisure platform, then materialized the obtained outcomes into 11 assessment criteria (figure below). We used these criteria to frame the review of 12 candidate tools which consisted in off-the-shelf prototyping tools in the following application domains: home automation, IoT, interactive user experience, education of programming and industrial process control.

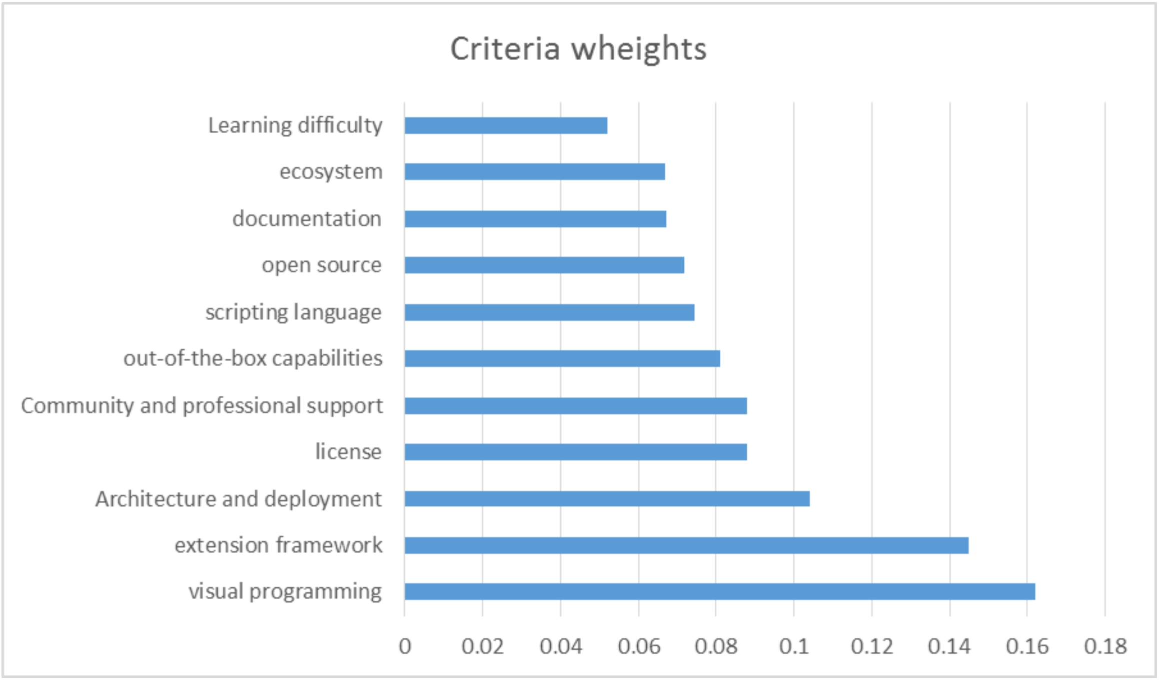


Figure 5: Weighted selection criteria

2- Assessment of candidate tools and overall rating

To assess the candidate tools we used the HAM scale with discrete values in [-8, -6, -4, -2, 0, 2, 4, 6, 8]. Upper and lower bounds of the scale was mapped to semantic interpretations (as specified in the table below) in order to ensure consistency in assessment of the tools.

Criteria/ interpretation	Extremely low (-8)	Average (0)	Extremely high (+8)
Out of the box capabilities for rapid prototyping	None of the elicited requirements.		All of the elicited functional and non-functional requirements.
Visual programming	Provides no visual programming UI.		Contextual palette of widgets to drag-and-drop, and various types of linkers.
Scripting language	No support.	Specific (non-standard) scripting language.	Popular and standard scripting language (JavaScript, Python..)
Extensions framework	No extensions framework.		Module based framework with support of inheritance mechanism and

			standard programming language.
Ecosystem	Vendor products only.	Vendor and partners' products.	Products compliant to open standards.
Open source	No access to source code.		Source code is available to the public and hosted on Git like repository.
License	Proprietary license under which derivative work is not permitted.		Free software license under which linking, distribution and modification are not restricted.
Deployment and architecture	Monolithic architecture, non-standard runtime, deployment limited to one platform.	Combination of the 2 extremes.	Component based, multi-platform deployment, standard and scalable runtime.
Community and professional support for third party product integration	No professional support, non-existent or very weak community.	Professional support or active community.	Professional support and very active community.
Learning curve for developing extensions	Difficult (more than 3 weeks) for a generalist software development practitioner.	Average (2 weeks) for a generalist software development practitioner.	Easy (1 week or less) for a generalist software development practitioner.
Documentation	No documentation.	Basic documentation.	Books have been published on the matter.

Table 6: Mapping of MAH scale to criteria semantics

The following three diagrams show actual HAM scale values attributed to each pair of (tool, criteria).

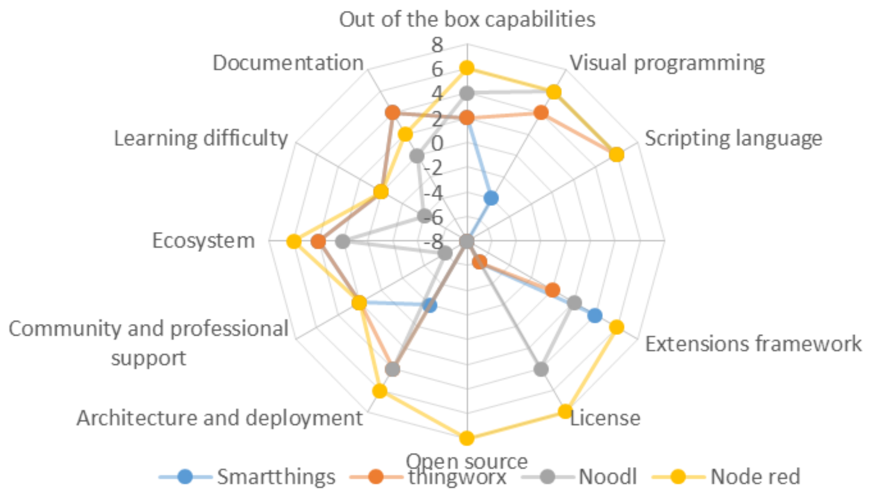


Figure 6: Tools assessment (part 1)

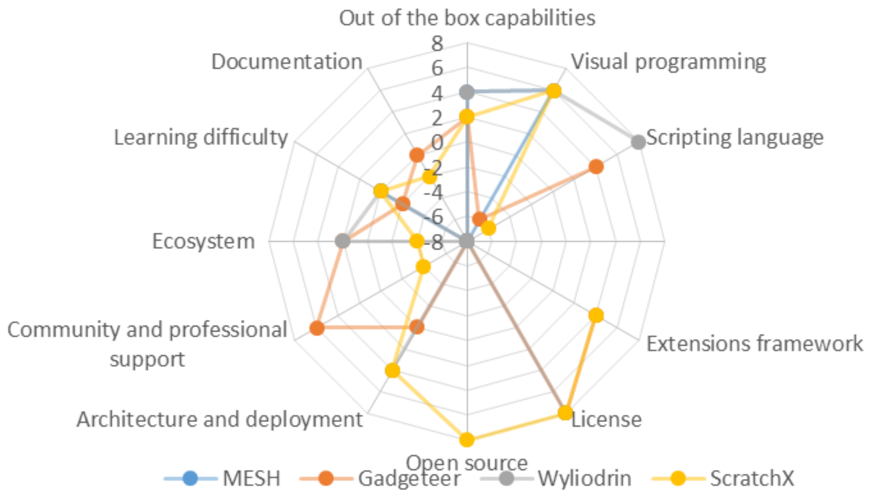


Figure 7: Tools assessment (part 2)

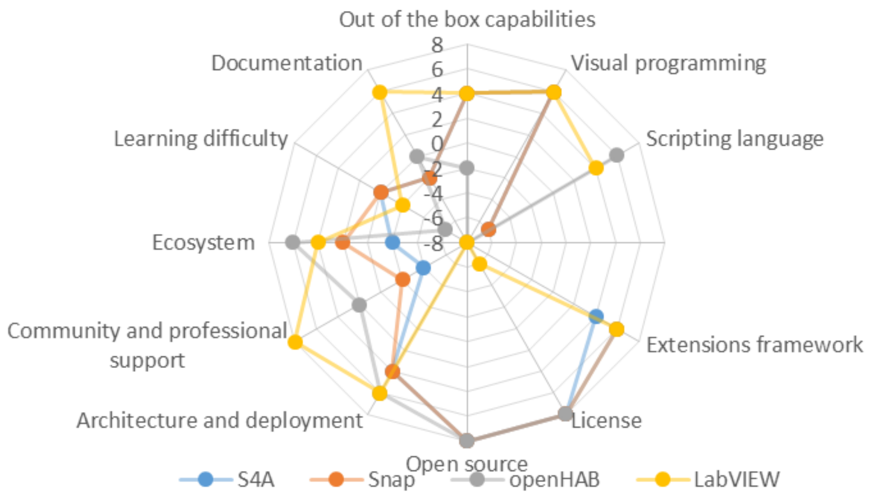


Figure 8: Tools assessment (part 3)

In the last step, we calculated the rating of each assessed tool (see chart below) using the HAM algorithm. “Node red” scored the highest rating, and hence we selected it as the visual programming and execution engine components of SHRPE. Node-red fulfills most of the functional and non-functional requirements, in addition to having a very active community, rich components library and an acceptable level of documentation describing the extension mechanism.

From a user perspective, integrating Node-red into SHARPE, allowed the platform to offer users the ability to visually design smart home scenarios by dragging/dropping widgets (or nodes) from a palette. Each node can be subsequently configured via a corresponding UI form, and links between nodes can be drawn in order to construct a logical flow through which messages are exchanged.

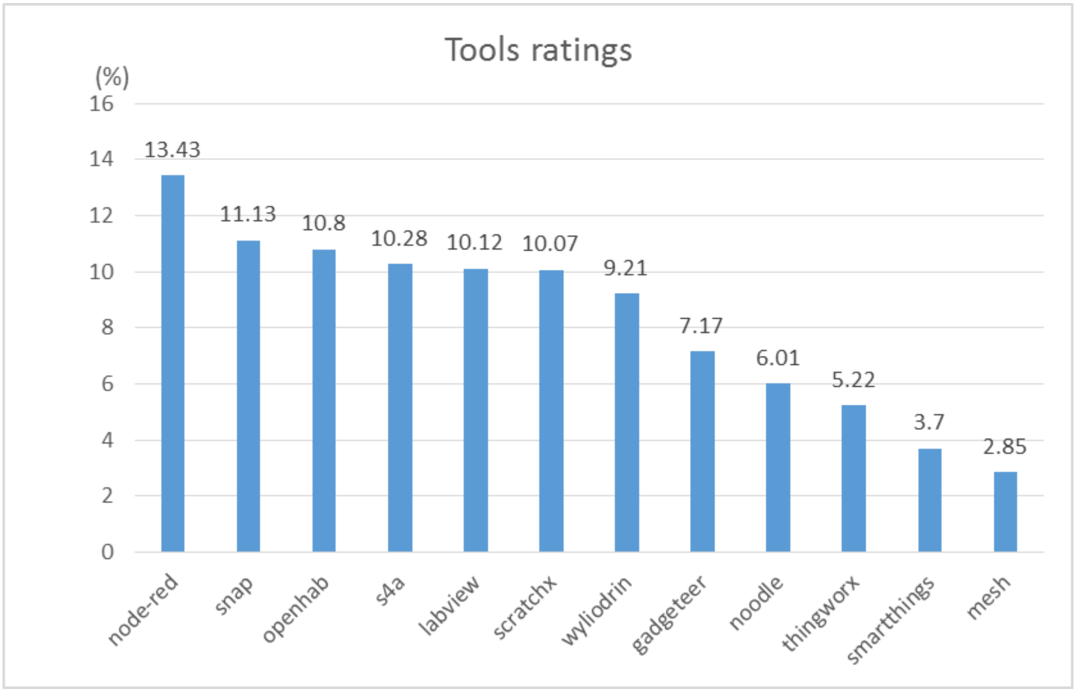


Figure 9: Ratings of reviewed tools

5.5 Integration architecture

5.5.1 Introduction

In this chapter, we introduce our proposal of a software architecture design for the Smart Home Rapid Prototyping Environment (SHRPE). The proposed design should serve as a foundation towards future architecture-specification work, hence, it highlights basic terminology and concepts – helpful in framing discussions –, and put them into the context of our present project. Although various aspects shall be considered when defining such artefact, our investigation has been scoped to cover the following two facets, with preference to adopt and adapt established industry standards whenever it applies, and consider specific constraints of VerisureIQ platform:

- Domain model.
- Foundation of integration architecture.

5.5.2 Application layer protocol

When reviewing state-of-the-art application layer protocols for the IoT, we were able to identify two major standardized protocols, MQTT and CoAP. Each of them is based on a different communication model and uses different but complementary semantics – MQTT is based on a connection oriented publish/subscribe pattern and message centric approach, CoAP is based on a connection-less client/server pattern and resource centric approach – with little overlap, if we consider the CoAP "observe" method, which emulates the publish/subscribe pattern.

At this point of the design process, we decided to embrace the CoAP protocol due to the rich ecosystem of related standards covering the scope of our investigation and beyond. However, MQTT will remain in our area of interest and will be used to fulfil a secondary role in the architecture of our functioning proof of concept, described in the next chapter.

In the following subsections, we will be referring to a collection of standard or draft specifications which whether extend the main CoAP specification [38], or conversely represent one of its dependencies. Our work consisted in selecting relevant parts from these specifications, and hence adapting and simplifying useful constructs, in order to satisfy our previously defined functional and non-functional requirements.

The aim of the remaining is to convey to the reader, our process of designing the application layer. The design outcome, may contribute to the ongoing discussions in academia as well as in IoT and Home Automation industries. In particular, projects currently investigating novel approaches to system transformation practices, towards realizing the WoT vision.

5.5.3 Domain model or device representation

In accordance to what is described in the objectives section of this report, our design process entails the investigation and definition of a conceptual model overseeing the problem domain. This model shall satisfy the criteria of technology agnostic nature, and suggests structural and behavioral features of the constituting elements.

Choosing a starting point for the design work, was a challenge, considering the variety of perspectives whose seem to be adequate candidates, for instance perspectives centric to: user, context, device, or system, are all possible options. Therefore, the first step was to decide on following one of these paths.

Considering the consolidated results of workshops 1 and 2, during which, smart-home scenarios have been suggested and visually formalized by participants, the majority of produced artefacts, had their associated mental-models converging to the device-centric approach. More precisely, a collection of independent physical devices and logical services interacting through messages exchange. This perspective implies the emergence of a system whose boundaries and lifetime, are tied to the time-window in which a given scenario executes. Three basic communication semantics were identified: event notification, request for information and call for action. Hence, our first attempt for a domain model, illustrated in the figure below, equips every device with respectively a collection of: capabilities (command, query), properties, and events (that could be emitted).

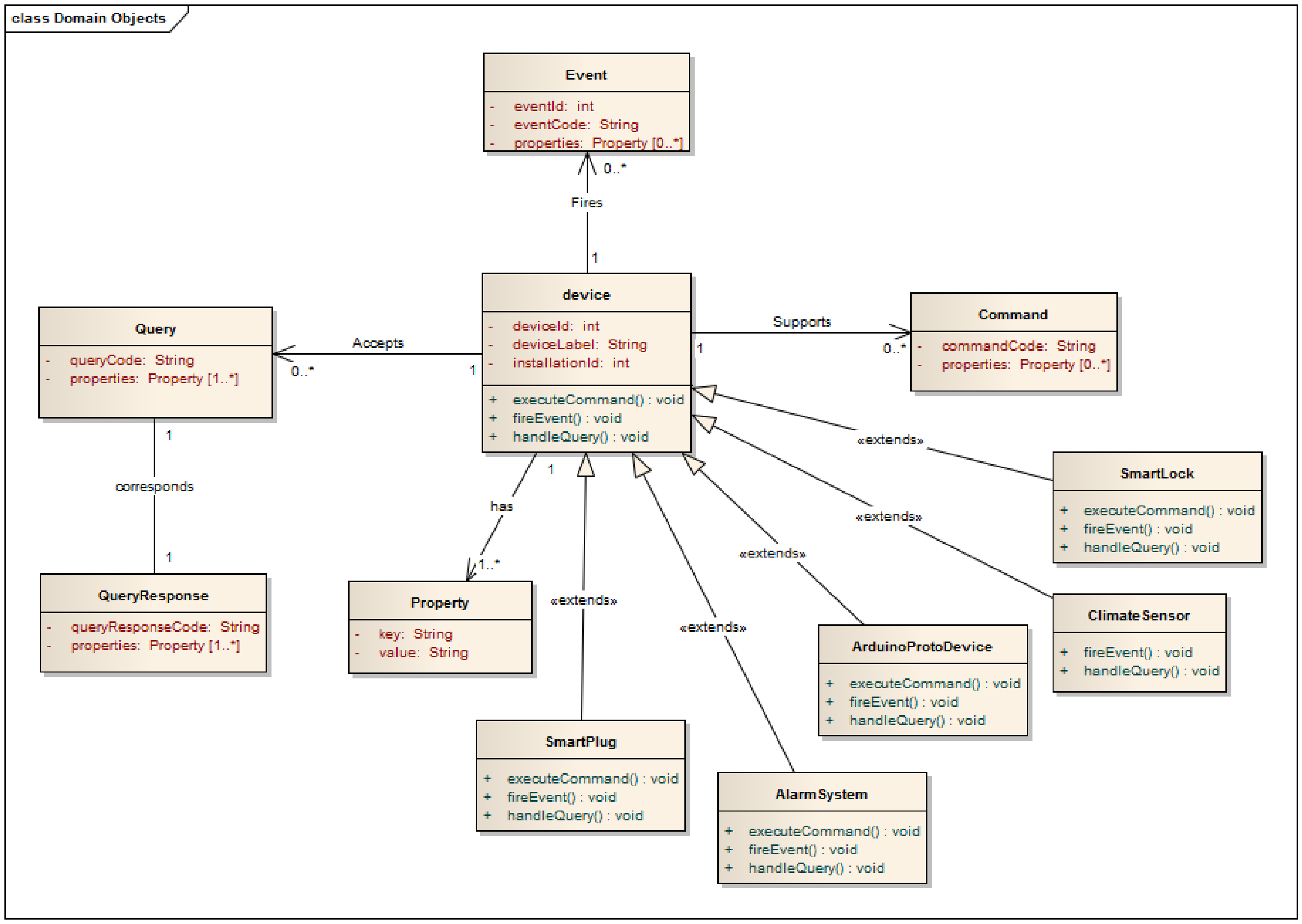


Figure 10: Domain model preliminary proposal

From another side, consideration of the CoAP protocol and its related specifications, for instance, OMA LWM2M [39], has resulted in aligning the former proposal with these standards. Therefore, we substitute the former with another device-centric domain model, where all attributes are represented as resources.

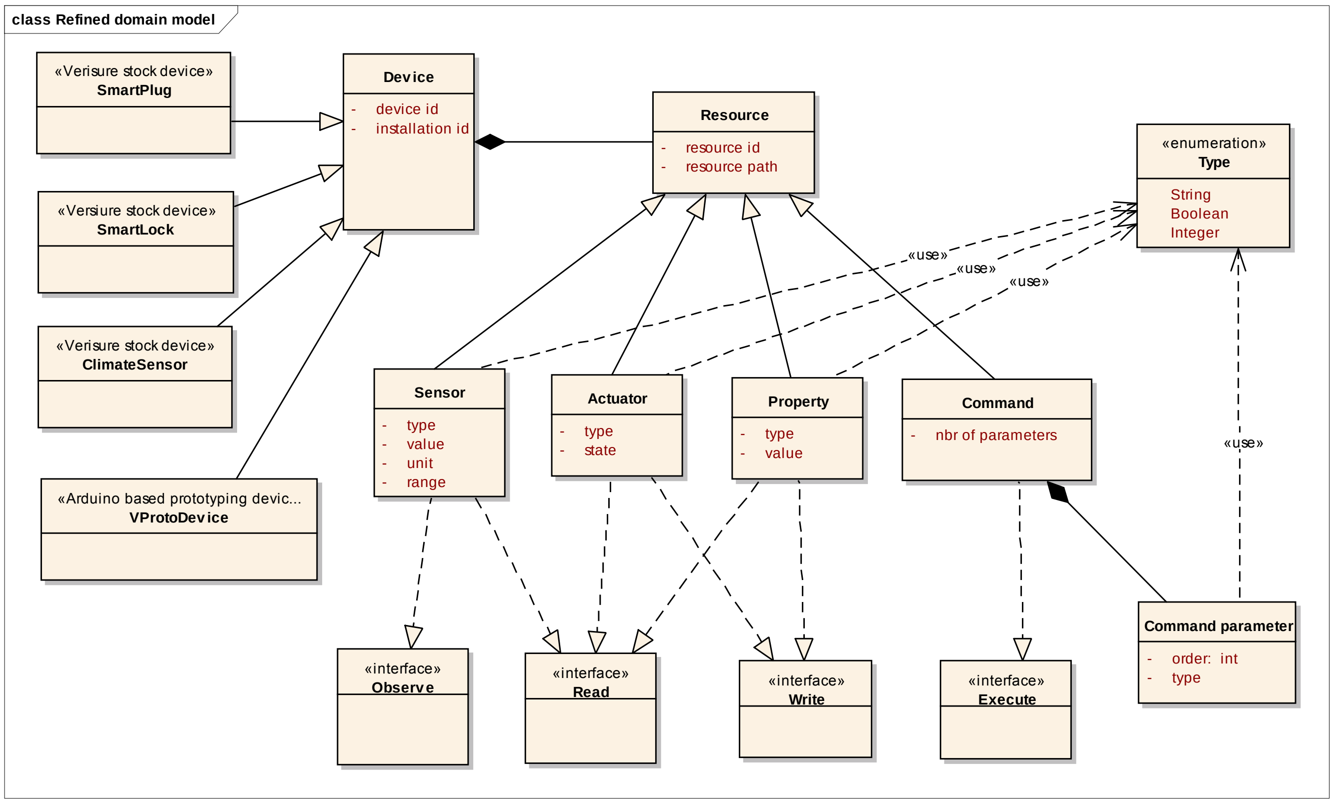


Figure 11: Domain model refined proposal

The diagram shows how a physical device consists in a collection of instances of reusable (or template) resources. This design suggests a simplified version compared to OMA LWM2M model, where the original model adds another level to the hierarchy, a container called “object” regrouping resources. Such simplification decreases complexity and allows optimization of last mile payloads, by the latter we mean: packets transiting between a connected device and the first non-constrained network node at the edge of the platform. Other identified technical constraints are discussed further in a separate sub-section.

In our model, inspired by IPSO reusable smart objects specification [40], we define four classes of resources. Each type carries a particular semantic and supports a subset of allowed operations: read, read + observe, write, and execute. The following table regroups these definitions.

Resource name	**Supported operations**	**Type**	**Description**
Sensor	Read + observe	String, boolean, integer	Represents a sensor or more generally, an input (Temperature sensor, button…). The value of a sensor can be read on demand or observed for changes via its corresponding resource. Example: a temperature sensor can be represented by an integer sensor resource, where the **range** of possible values is [-100,+350] expressed in Celsius degrees **units**.
Actuator	Read, write	String, boolean, integer	Represents an actuator (LED, relay…). An actuator is by overriding its state, the type or format of the state depends on the nature of corresponding actuator. Example: an LED can be represented by a boolean actuator resource, the value of true (or 1) turns it ON, and the value of false (or 0) turns it OFF.
Property	Read, write	String, boolean, integer	Represents a static information embedded in the device. The value of a property can be read and written. Example 1: the version of the prototype can be represented as an integer property resource. Example 2: the code-name of the prototype can be represented as a string property resource.
Command	Execute	Array of string	Represents a parametrized complex logic which the device can perform, we can see it as an executable macro function. Each time a command is called, all required arguments or parameters shall be included within the call. Example: a device equipped with a step motor, exposing a command to control how it spins. Parameters could be: direction [“clockwise”, “counterclockwise”] and speed [0,10].

Table 7: Domain model - definition of resources

5.5.4 Foundation for target integration architecture overview and components

The following diagram shows our proposed architecture foundation to perform software integration of SHRPE with VerisureIQ. Components of the architecture are described in the following sub-sections.

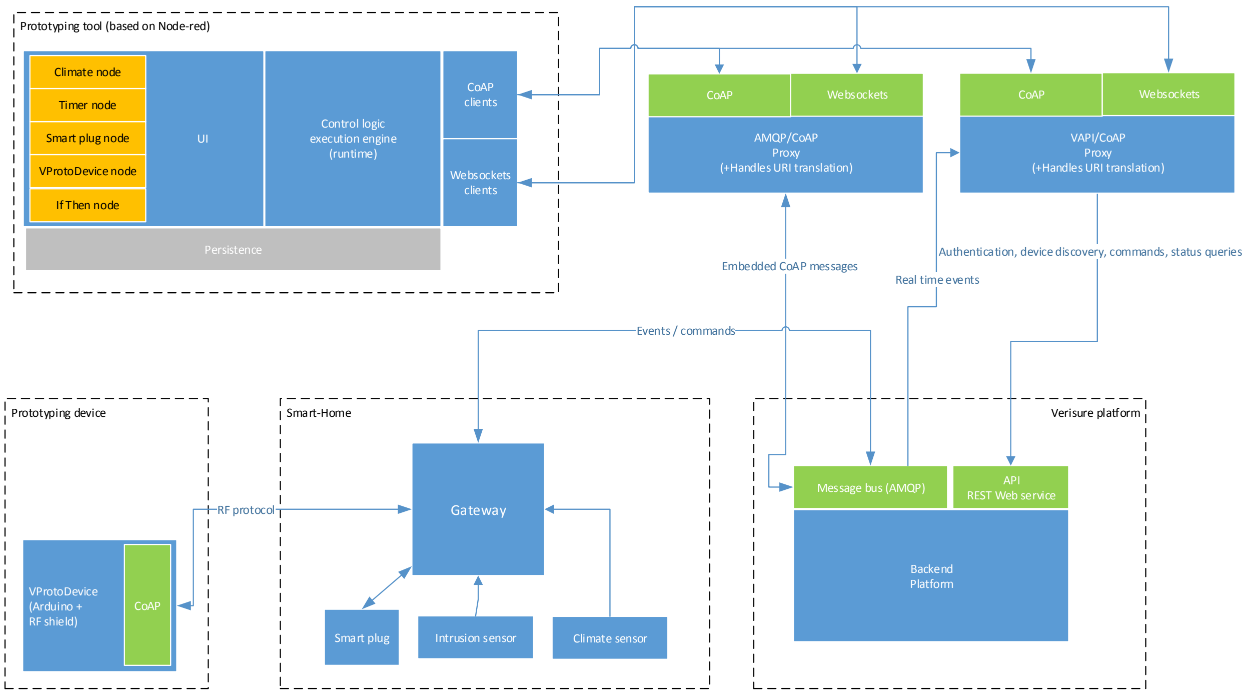


Figure 12: Architecture foundation of SHRPE

5.5.4.1 VerisureIQ platform and smart home installations

A description of this macro-component was presented in (section 5.2).

5.5.4.2 Prototyping device

SHRPE use cases are not limited to visually programming scenarios over existing commercial devices in VerisureIQ platform, like SmartPlug, ClimateSensor and others. But it expands the prototyping spectrum to bring physical device prototypes into the ecosystem. These devices which we named VProtoDevice, consist of an Arduino Mega board, on top of which is installed a radio frequency (RF) shield compatible with gateways produced by Verisure. Design and implementation of the shield is done in the scope of two other students' joint thesis.

The software part, known as firmware, shall implement a CoAP server template, in addition to a native library assuring datagram transport over the proprietary RF protocol.

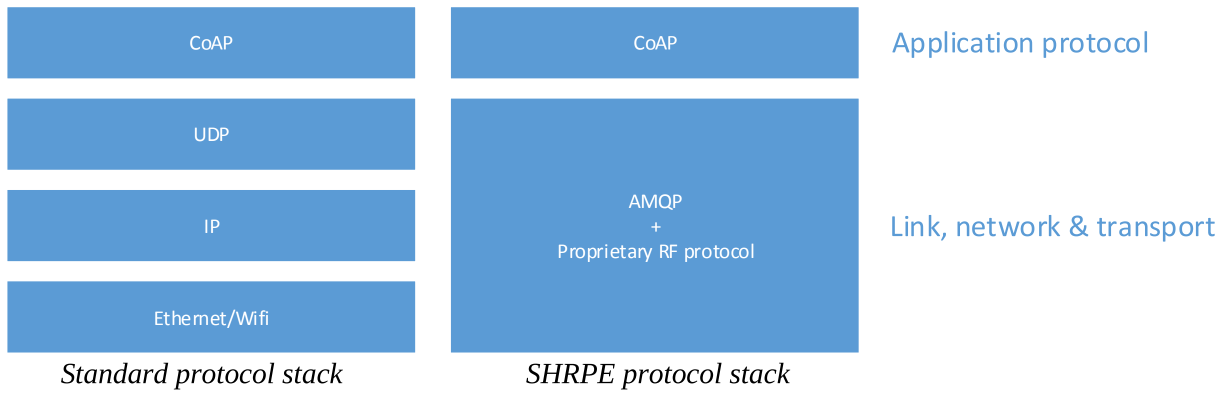


Figure 13 : CoAP protocol stack - standard vs SHRPE specific

The figure above highlights differences between the standard protocol stack, on top of which CoAP traditionally works, and, the non-conventional configuration used in SHRPE to support the integration of physical prototyping devices. Initially designed to operate over UDP at the transport layer, CoAP needs a custom made binding for it to operate over VerisureIQ platform. In this case, the RF protocol is fulfilling the link and transport layer, whereas the network layer is formed by the gateway and the backend of VerisureIQ (business logic with AMQP).

A more comprehensive protocol stack is presented in the next figure, it covers in addition, the service or API layer and provides a data model facility for application-level message exchange.

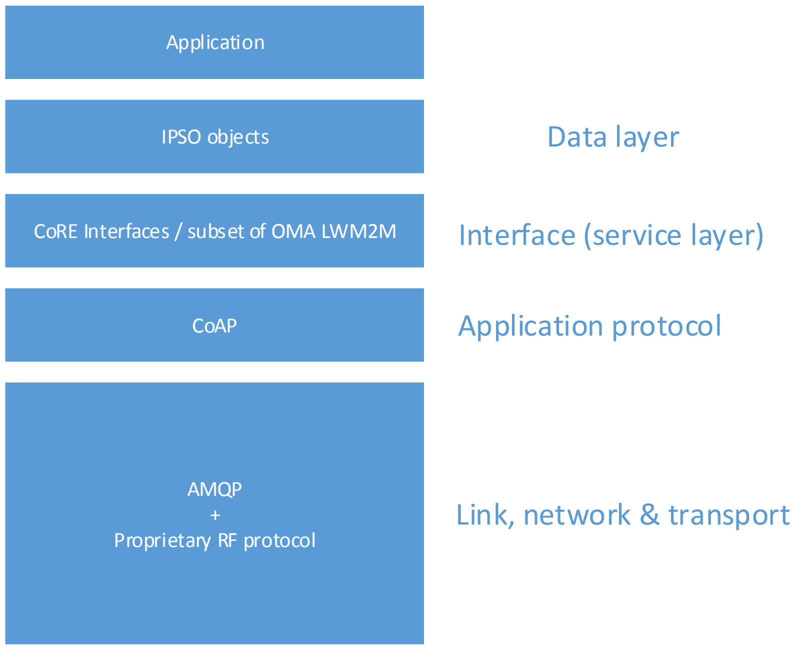

Application
IPSO objects
Data layer
CoRE Interfaces / subset of OMA LWM2M
Interface (service layer)
CoAP
Application protocol
AMQP
+
Proprietary RF protocol
Link, network & transport

Figure 14: Detailed protocol stack in SHRPE

5.5.4.3 Prototyping tool

In SHRPE architecture, the prototyping tool fulfills two main functions. First, a front-end facility allowing users to visually express smart-home scenarios including commercial and prototyping devices connected to the gateway, as well as external services. Second, an execution engine responsible for running deployed scenarios and perform necessary communications with the VerisureIQ platform.

In (section 5.4) we selected Node-red, the IoT oriented visual programming tool which serves as the foundation of this component. The choice was motivated by the fact that Node-red implements required features to fulfill the two main functions mentioned earlier. However, it is necessary to design SHRPE specific extensions (UI and integration logic sub-components), in order to assure interfacing with VerisureIQ platform and deliver relevant visual widgets. These widgets (or nodes) represent abstractions of the connected devices, yet implementing interaction primitives as defined in the domain model (section 5.5.3) and application protocol (section 5.5.2).

According to Node-red logical structure, a node can be composed of, input, internal logic, and outputs. Inputs and outputs are interfaces receiving, or emitting, messages

(JavaScript objects) from, or to, other nodes in the flow. Internal logic has the ability to manipulate these messages and also to access system and network resources via the runtime environment (Node.js). In our design, each connected device is represented by its corresponding SHRPE node, and communication with SHRPE middleware takes place in the internal logic bloc using CoAP protocol. From a conceptual point of view, connected devices can be considered as CoAP servers, and SHRPE nodes (node-red nodes we have designed, details are presented in chapter 8) as CoAP clients.

5.5.4.4 CoAP proxy

In the emerging CoAP ecosystem, CoAP proxies – seen as networking nodes – enact an essential role in enabling interconnection of CoAP-based IoT platforms with other heterogeneous platforms, for instance modern or legacy corporate systems – which conventionally expose interfaces over different yet established communication protocols.

Despite its importance, this topic has gained limited attention in literature and among all draft proposals related to CoAP specification. In their work, Ludovici and Calveras [41] contributed to the ongoing discussion with a comprehensive proxy design, supporting intensive event driven IoT applications. The proposed design enables communication between CoAP devices and web applications mediated by a specific type of proxy; we call this class of proxies “cross-protocol” as defined in [42], its main role is to perform 2-way translation between two protocols, for instance CoAP to HTTP.

In this project, interest is turned to a specific class of applications, SHRPE is a centralized event driven IoT application, in which near real-time data being pushed by CoAP capable devices is consumed and processed by a centralized application server.

Design guidelines introduced above have been leveraged to support engineering the Middleware layer of SHRPE. The result consists in a design comprising two CoAP proxies, illustrated in Figure 12. Each providing connectivity capabilities available to serve a sub domain of devices in the platform, Verisure commercial devices as opposed to Arduino based prototyping devices (VProtoDevice).

From an architecture point of view, the suggested proxy specification listed below, defines two domain specific gateways, such components are in addition responsible for handling internal logic which is specific to Verisure platform. Therefore, it should not be confused with platform agnostic proxy design, which operates only on protocol level.

Proxy name	Protocol mappings	Rational	Requirements
VProtoProxy (Dedicated to VProtoDevices)	CoAP-AMQP	A VProtoDevice is only accessible through publish/ subscribe on the message bus	- Expose CoAP interface - Tunnel CoAP through AMQP in the 2-ways
ComDevProxy (Dedicated to Verisure commercial devices)	CoAP-HTTP CoAP-AMQP	Commercial devices are accessible via the Verisure REST API (VAPI). But real-time observation of resources can only happen through the message bus.	- Expose CoAP interface - Translate CoAP to REST - Support CoAP resource observation with input from AMQP

Table 8: CoAP proxy details

<u>URI scheme mapping</u>

Depending on the design and operational environment of proxy components, certain situations require the definition of customized URI schemes. Such schemes offer the possibility to accommodate non-standard pieces of information, which are necessary for realizing new protocol binding. Proxies take advantage of embedded information in URIs, to perform their role in translating and routing messages which transit through their interfaces.

When applying the above considerations to SHRPE, the need for customized URI scheme arises. A published draft “CoAP Communication with Alternative Transports” [43] conveys practical recommendations on the topic in relation to CoAP. Thus we propose a URI scheme which embraces the internal hierarchy of Verisure platform, it is composed of the following:

- Authority component: coap-shrpe
- Root path component: <installation_id>
- Second level path component: <node_id>

URI template: coap-shrpe://<installation_giid>/<node_id>

5.5.4.5 RF protocol limitations

In Verisure platform a proprietary RF protocol is used to assure communications between the gateway component of a home installation and other installed smart devices, for instance, smart plugs, climate sensors. Referring to our proposed architecture, the RF protocol was selected to perform the same role in integrating Arduino based prototyping devices (VProtoDevices) within SHRPE platform.

The considered RF protocol provides a connectionless and unreliable link layer, with a packet payload limited to 35 Bytes. Our proposal suggests transporting CoAP messages as payload of these packets. Although CoAP was designed as a compact protocol for constrained devices, we were required to define a SHRPE specific upper bound on the size of exchanged CoAP messages, therefore overriding the standard upper bound set by the CoAP specification [38]. A similar situation was briefly discussed in CoAP specification which recommends implementing on the server side an error handler as per the following statement: the server should reply with “Request Entity Too Large” (code 4.13) if received a request exceeding its implementation upper bound.

To minimize the chance of such scenarios to occur, we argue for a solution to reduce the footprint of exchanged CoAP messages, while keeping their payloads human readable. For this we use the CoRE Link Format [44] expressed in JSON notation and then compress the message payload using CBOR [45] prior to transport over the RF protocol. Such compression method was proposed in the draft “CBOR Equivalents of CoRE JSON Formats” [46].

5.6 SHRPE prototype

Based on our work presented in the previous section (5.5) and which involved system analysis and architectural work, we defined a narrower scope to implement and evaluate a functioning prototype. The scoping was performed through backlog prioritization in line with the Agile methodology. We used the “walking skeleton” strategy [47] to select a set of requirements which ensued the implementation of certain segments of the foundation architecture. This strategy assured the relevance of the prototype evaluation to mainly answer the technical feasibility question embodied by our main research question and hypothesis.

5.6.1 Prioritized backlog

For this prototype we adopted a product vision which focuses on enabling interaction across smart-home devices (or elements) belonging to the 3 different categories as described in the table below. Therefore, our definition of the "walking skeleton" from SHRPE perspective, was to succeed in supporting one type of device or service from each category and to demonstrate an interaction scenario using the implemented prototype.

Element category	Verisure stock device	Prototyping device	External API
Supported by the prototype	SmartPlug	Arduino with Wi-Fi shield	Twitter

Table 9: Element categories in SHRPE

Here we list the retained requirements out of the initial list in (chapter 4), and include new ones identified further in the process.

FR1: Drag and drop visual representations of smart-home devices

FR2: Drag and drop visual representations of external services

FR3: Drag and drop visual representations of basic control logic

FR5: Express scenarios through visually linking entities mentioned in FR1, FR2 and FR3

FR6: Deploy and execute scenarios

FR7: Export a scenario after finishing its design

FR8: Import a saved scenario

FR9: Support of a popular scripting language

FR10: Drag and drop visual representations of physical prototyping devices

The following table lists the requirements ranked according to priority (from higher to lower), and specifies the gap and scope in the context of SHRPE prototype.

Priority	Requirement	Action to fill the gap	Scope

10	FR1	Implement node-red extensions	SmartPlug device
10	FR10	Implement node-red extensions	Arduino device using CoAP protocol over Wi-Fi
10	FR3	Implement node-red extensions	Commands, query and filter nodes for SmartPlug and Arduino
10	FR5	Fulfilled by basic Node-red features	N.A.
10	FR6	Fulfilled by basic Node-red features	N.A.
20	FR2	Fulfilled by basic Node-red features	N.A.
30	FR9	Fulfilled by basic Node-red features	N.A.
40	FR7	Fulfilled by basic Node-red features	N.A.
40	FR8	Fulfilled by basic Node-red features	N.A.

Table 10: Prioritized backlog and gap analysis

5.6.2 Architecture overview of the prototype

Architecture of SHRPE prototype was based on the “foundation for target architecture” presented in (section 5.5.4). As the following diagram shows, proxy components are not covered, and usage of CoAP protocol is limited to the Arduino device. Another simplification consists in using the available REST API to control SmartPlug devices, real-time notifications were simulated by a combination of a polling mechanism (sending REST requests) and a MQ Telemetry Transport (MQTT) message broker.

Although we proceeded with a number of simplifications from a networking and message routing perspective, we maintained the communication semantics of CoAP (read resource, observe resource, write to resource) at two levels, the first within the Arduino based device which implemented an actual CoAP server managing 3 resources (sensor, actuator and property); the second within implemented extensions of node-red, that is SHRPE custom nodes (see next section) which users can utilize in node-red visual editor.

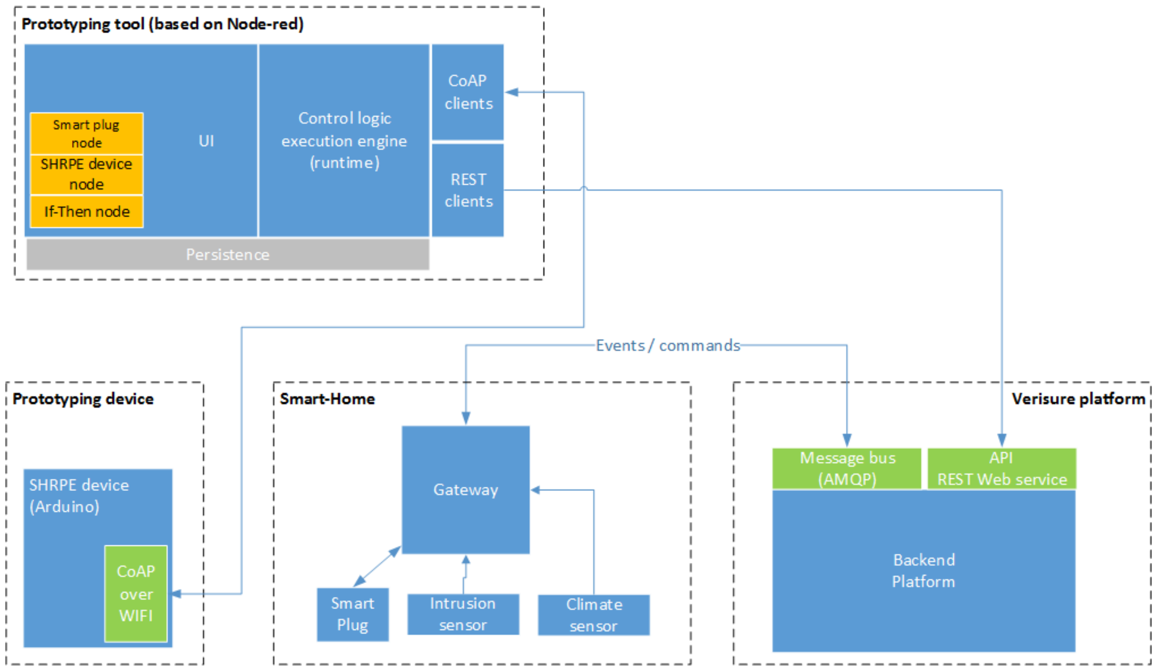
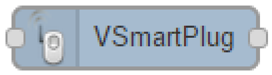
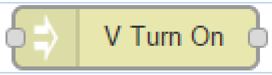
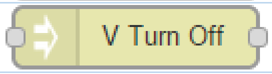
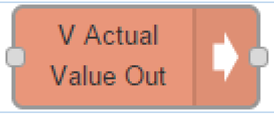


Figure 15: Architecture overview of SHRPE prototype

5.6.3 SHRPE nodes

To implement SHRPE prototype, we developed a set of node-red extensions in order to fill the feature gaps identified in (section 5.6.1). These node-red extensions are custom nodes materializing SHRPE extensions. In the following table we describe the functionality of each implemented custom node.

Extension	Output compatible with input of	Functionality
 VSmartPlug	V Actual Value Out V Event Out	Perform outbound/inbound message translation and routing to/from a SmartPlug device using the Verisure platform.
 V Turn On	VSmartPlug V Actuator In	Outputs the command to turn ON a SmartPlug or an actuator attached to a VProto device.
 V Turn Off	VSmartPlug V Actuator In	Outputs the command to turn OFF a SmartPlug or an actuator attached to a VProto device.
V Get Status	VSmartPlug V Sensor In V Actuator In V Property In	Outputs a request to read the current status of a SmartPlug or a sensor attached to a VProto device.
 V Actual Value Out	N.A.	Outputs the current status (response to a read request) of a SmartPlug or a sensor attached to a VProto

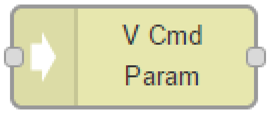
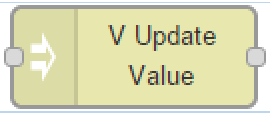
	N.A.	Outputs the new status (if changed) of a SmartPlug or a sensor attached to a VProto
	V Actual Value Out V Event Out	Performs outbound/inbound message translation and routing to/from an Arduino device running a CoAP server
	VProtoDevice	Performs routing of read and write messages to the corresponding (configured via UI) actuator resource in a VProtoDevice.
	VProtoDevice	Performs routing of read messages to the corresponding (configured via UI) sensor resource in a VProtoDevice.
	VProtoDevice	Performs routing of read and write messages to the corresponding (configured via UI) property resource in a VProtoDevice.
	VProtoDevice	Performs routing of command execution messages to the corresponding (configured via UI) command resource in a VProtoDevice.
	V Command In	Outputs a formatted command parameter based on input message. The output is used to parametrize the execution of a command resource in a VProtoDevice. One command can support up to 5 parameters.
	V Property In	Outputs write message to update a property resource value in a VProtoDevice.

Table 11: Extension nodes of SHRPE prototype

6 Results

According to DSRP, the last step in our research process is to evaluate the two aspects relevant to our research questions. First, the main research question was addressed by functional evaluation of the implemented SHRPE prototype, to measure conformity against specified requirements. Second, our research sub-question was addressed by conducting a semi structured questionnaire comprising quantitative elements, to measure adequacy of the UCD approach in capturing user needs within such project setting.

We performed this evaluation in a third workshop in which we demoed the prototype and conducted user testing to address the questions mentioned above. Out of all the participants who contributed in past workshops, 7 took part in this evaluation which followed the agenda below:

Presentation of Node-red and SHRPE prototype [20 minutes]

Session 1 (group 1, 4 participants): scenario based testing [30 minutes]

Session 2 (group 2, 3 participants): scenario based testing [30 minutes]

Questionnaire (all participants)[10 minutes]

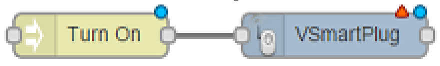
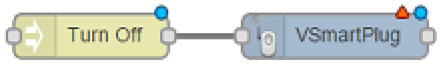
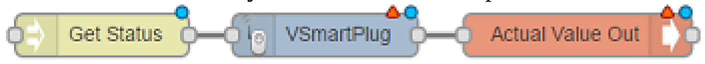
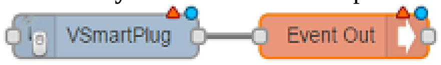
Plenary discussion and closure [30 minutes]

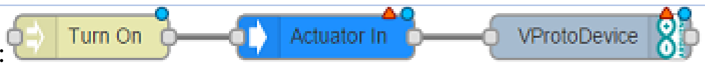
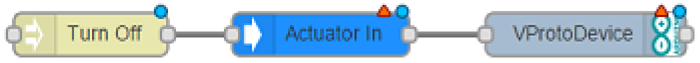
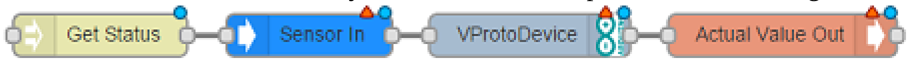
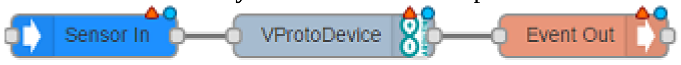
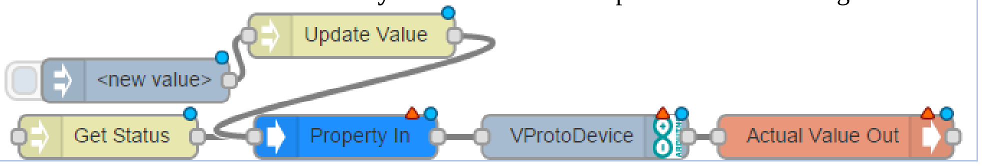
6.1 Scenario based testing

Each testing session started with a detailed explanation of custom Node-red nodes implemented in the scope of this project, then the actual testing was performed collaboratively with our involvement and assistance, due to limited time available for making participants familiar with the tool.

6.1.1 Basic scenarios

First, basic scenarios were tested accordingly to the test plan below. The table shows definition and result of each performed test, and regroups tests in two main categories following the nature of the covered physical device: the first category comprises stock devices in Verisure platform, while the second category encompasses Arduino based prototyping devices.

Category	Test definition and result
Verisure stock device	• **Test reference:** SmartPlug/ toggle on off • **Test purpose:** SmartPlug device should support "Turn ON" and "Turn OFF" commands which can be triggered as part of a user defined scenario in node-red. • **Software artefacts covered by the test:** 3 SHRPE specific node-red widgets: Phase 1:  Phase 2:  • **Test pre-requisites:** Use a SmartPlug and a desk lamp. Plug the desk lamp into the SmartPlug. Turn the switch of the desk lamp to ON position. Phase 1**:** Manually turn the SmartPlug to OFF using the built-in button. Phase 2: Manually turn the SmartPlug to ON using the built-in button. • **Expected result:** Phase 1: After the "Turn On" command is triggered, the SmartPlug is switched to ON state, observed by the desk lamp lit on. Phase 2: After the "Turn Off" command is triggered, the SmartPlug is switched to OFF state, observed by the desk lamp lit off. • **Test result:** Success
	• **Test reference:** SmartPlug/ current_state • **Test purpose:** SmartPlug device should support and respond to the queries for current state which can be triggered inside a user defined scenario in node-red. • **Software artefacts covered by the test:** 3 SHRPE specific node-red widgets:  • **Test pre-requisites:** Use a SmartPlug and a desk lamp. Plug the desk lamp into the SmartPlug. Turn the switch of the desk lamp to ON position. For step 1: manually turn the SmartPlug to ON using the built-in button. For step 2: manually turn the SmartPlug to OFF using the built-in button. • **Expected result:** After the "Get Status" command is triggered, "Actual Value Out" should output a value matching the SmartPlug state (in each step). • **Test result:** Success
	• **Test reference:** SmartPlug/ observe state • **Test purpose:** SmartPlug device should report state changes and make such event available inside user defined scenarios in node-red. • **Software artefacts covered by the test:** 2 SHRPE specific node-red widgets:  • **Test pre-requisites:** refer to the previous test (smart_plug_current_state) • **Expected result:** "Event Out" should output a value matching the SmartPlug state (in each step). • **Test result:** Success
Prototyping device	• **Test reference:** prototyping device/ actuator/ toggle on off • **Test purpose:** Arduino based prototyping device should support "turn ON" and "turn OFF" commands directed to actuators, which can be triggered inside a user defined scenario in node-red. • **Software artefacts covered by the test:** 4 SHRPE specific node-red widgets:

	Phase 1:  Phase 2:  • **Test pre-requisites:** Use the Arduino based prototyping device and an LED. Connect the LED anode leg to one of the digital outputs in the device. • **Expected result:** Phase 1: After the "Turn On" command is triggered, the LED switches ON Phase 2: After the "Turn Off" command is triggered, the LED switches OFF • **Test result:** Success
	• **Test reference:** prototyping device/ sensor/ read value • **Test purpose:** Arduino based prototyping device should support and respond to the queries for reading sensor value, which can be triggered inside a user defined scenario in node-red. • **Software artefacts covered by the test:** 4 SHRPE specific node-red widgets:  • **Test pre-requisites:** Use the Arduino based prototyping device and a push button. Connect the push button to one of the input pins and to the power pin in the device. For step 1: hold push the button. For step 2: release the button. • **Expected result:** After the "Get Status" command is triggered, "Actual Value Out" should output a value matching the button state (in each step). • **Test result:** Success
	• **Test reference:** prototyping device/ sensor/ observe state • **Test purpose:** Arduino based prototyping device should report state changes of an observed sensor and make such event available inside user defined scenarios in node-red. • **Software artefacts covered by the test:** 3 SHRPE specific node-red widgets:  • **Test pre-requisites:** refer to the previous test (prototyping device/ sensor/ read value) • **Expected result:** "Event Out" should output a value matching the button state (in each step). **Test result:** Success
	• **Test reference:** prototyping device/ sensor/ read value • **Test purpose:** Arduino based prototyping device should support and respond to queries for updating or reading property value, which can be triggered inside a user defined scenario in node-red. • **Software artefacts covered by the test:** 5 SHRPE specific node-red widgets:  Update Value <new value> Get Status Property In VProtoDevice Actual Value Out

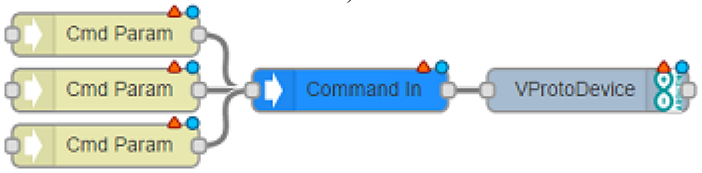
	• **Test pre-requisites:** Use the Arduino based prototyping device. • **Expected result:** After sequentially triggering the two commands "Update Value" and "Get Status", the widget "Actual Value Out" should output a value matching the message payload used to trigger "Update Value". **Test result:** Success
	• **Test reference:** prototyping device/ parametrized command trigger • **Test purpose:** Test limited to the node-red tool only, not implemented yet on the prototyping device side. "Command In" widget should collect received parameters until all 3 parameters are received, then generates a command message and make it available inside a user defined scenario in node-red. • **Software artefacts covered by the test:** 2 SHRPE specific node-red widgets ("Cmd Param" and "Command in"):  Cmd Param Cmd Param Cmd Param Command In VProtoDevice • **Test pre-requisites:** Not applicable. • **Expected result:** After sequentially triggering expedition of all 3 command parameters "Cmd Param", the widget "Comman In" should output a message payload with the following content: name of the command accompanied with an array of 3 elements, each element corresponds to the value of its respective command parameter. • **Test result:** Success

Table 12: Definition and results of basic scenario tests

6.1.2 Benchmark scenario

After testing basic scenarios, a more complex scenario was put into test, it covers three categories of devices or entities as defined in section 5.6.3 (Verisure stock device, prototyping device and external API). This scenario covers the prioritized requirements (section 5.6.1) which frame the scope of SHRPE prototype. The figure below is a screenshot of the scenario taken from node red tool, it shows how SHRPE nodes were used to realize the logic behind it.

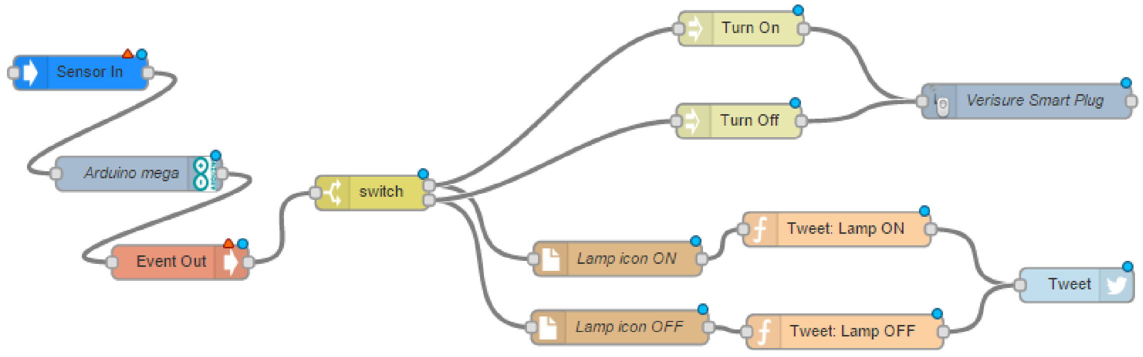


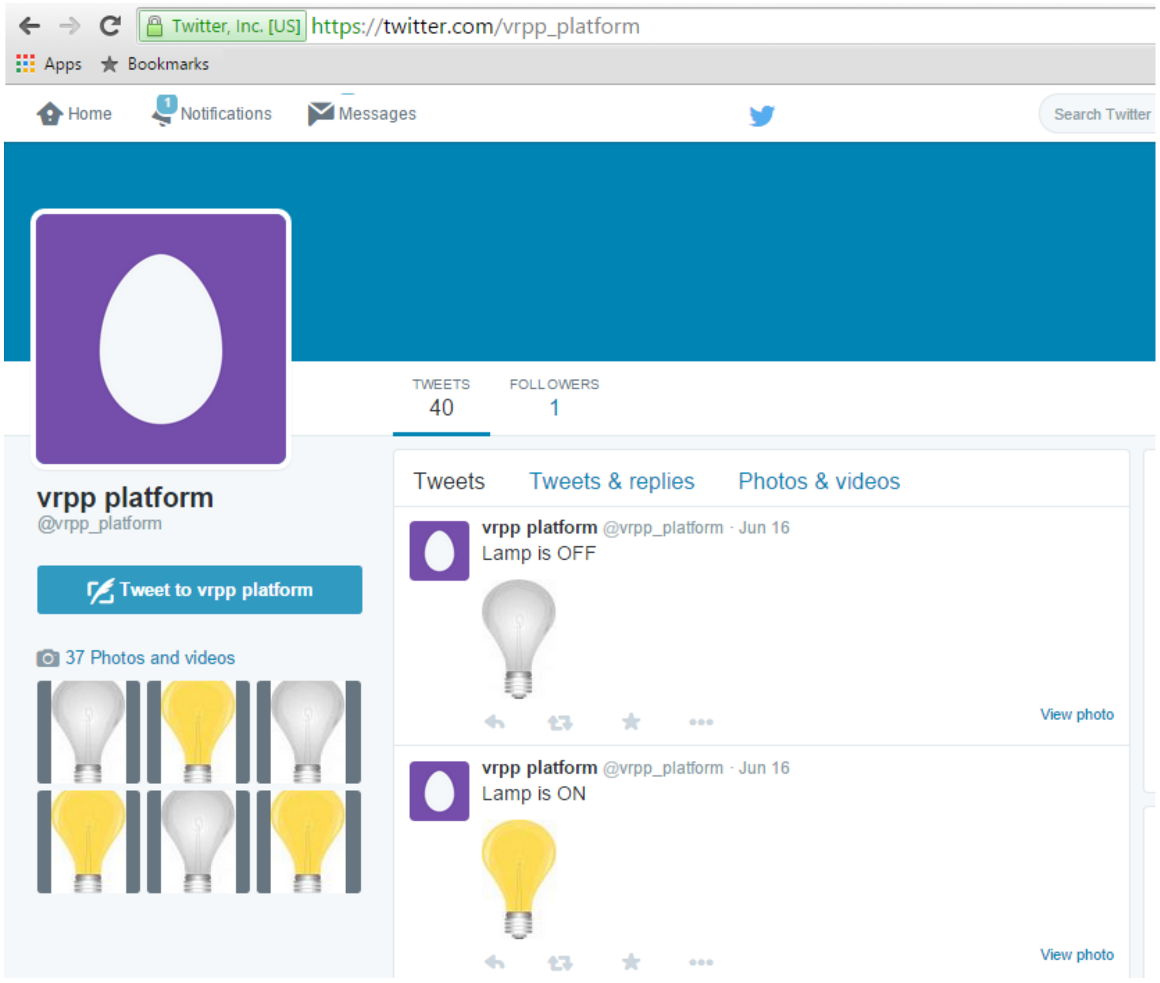
Figure 16: Benchmark scenario

Setting up our evaluation test required the following equipment:

- **Verisure stock device**: SmartPlug.
- Desk lamp switched ON and plugged into the SmartPlug.
- **Prototyping device**: Arduino Mega board (with WIFI shield) running CoAP server over UDP/WIFI.
- Push button wired into the Arduino board.
- **External API:** Twitter account (@vrpp_platform).
- Verisure gateway.
- WIFI router.
- Laptop running node-red (deployed on node.js server) with SHRPE nodes installed.

The scenario narrative says: *a participant is sitting and playing around with a button (attached to Arduino), when he/she presses the button, a lamp attached to the SmartPlug lights up, and at the same time a tweet is tweeted on @vrpp_platform (twitter account), containing a picture of a lamp (lit on) and a text status reading "Lamp is on!". Conversely, when the participant releases the button, the lamp turns off, and a tweet gets tweeted with picture of a lamp (lit off) and a text status reading "Lamp is off!".*

The result was a **success** confirmed by observation of both the lamp and tweets reacting to actions performed on the button, with an average latency of 3 seconds for the lamp, and 4 seconds for twitter. No special setup was utilized for measuring latency aside from a stop watch, since the purpose of this evaluation do not cover performance measurement.



vrpp platform @vrpp_platform · Jun 16
Lamp is OFF

vrpp platform @vrpp_platform · Jun 16
Lamp is ON

Figure 17: Tweets in reaction to push/release button

6.2 Evaluation of UCD adequacy

We remain in the context of our evaluation workshop, after having performed scenario-based testing, we turned to the second aspect in which we are attempting to assess adequacy of UCD approach to capture user needs in this project. This was done through an expert opinion questionnaire, designed as semi-structured and included quasi-quantitative input, throughout 5 questions we asked participants about features and behavior of the developed artefact, and whether it corresponds to their expectations.

However, we need to consider the risk of intrinsic bias in such method due to the inherent subjectivity of collected data.

In the bar-chart below, we present the results obtained from questions 1, 2, 4 and 5 (see Appendix III), and mention the number of respondents (N) in order to reflect representativity of each aggregated score.

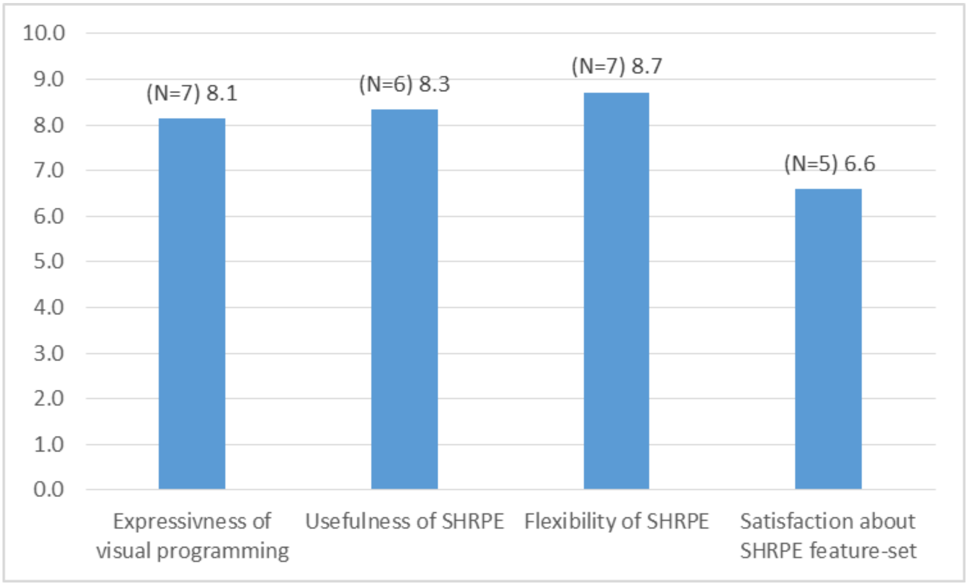


Figure 18: Average scores of expert opinion questionnaire

In reference to reported feedback from the first workshop (idea generation), during which participants were asked the same first and second questions, we acknowledge a sharp rise in how participants perceived the expression power of visual programming (Q1). Concretly, this was materialized by a very high average score 8.1 out of 10 in the final evaluation, compared to less motivated answers in the first workshop: ranging from a conservative yes (2 participants), an excited yes (2 participants) to explicit expression of doubt and uncertitude (2 participants).

Similarly, the second question (Q2) inquiring usefulness of such prototyping tool, and in particular SHRPE prototype, has seen a high satisfaction score of 8.3, confirming by that the marked unanimous initial impression of participants in the first workshop, where all 6 participants responded with a sharp yes.

7 Conclusion and future work

The limits and direction as stated in our research questions implied significant influence on outcomes, mainly opinions of participants as well as our own decisions which we undertook during the whole period of working on this thesis, and at several points of the design process. However, from DSRP perspective, this is compliant to its principles which instruct the researcher to force certain parameters in input then assert feasibility through measuring and assessing outcomes. Although it exhibits some similarities to experiment procedures, DSRP tolerates the usage of macro level, holistic approaches, to influence independent variables.

According to our thesis results, we confidently confirm that the UCD approach provides relevant insights to user needs and functional requirements, when used in the context of projects aiming at enhancing academic and professional research in the specific area of smart-home service innovation. Therefore our hypothesis of the research sub question is accepted.

From a technical perspective, applying principles of software integration with the help of a set of open source tools respectively open standard protocols, namely Node-red, Arduino and CoAP, showcased the opportunity to open up closed platforms, and bridge the current gap towards emerging WoT ecosystem. In this thesis we designed, implemented and tested a proof of concept, covering essential parts of the defined architecture foundation for SHRPE. The success of this first step proved the validity of the suggested approach in the hypothesis of our main research question and therefore this hypothesis is accepted.

In our thesis we did not investigate the impact of using the SHRPE tool on the workflow and innovation process of target users. This gap represents a relevant entry point for future work which is centered on the process enhancement aspect of rapid prototyping platforms for the smart home.

Appendix I: Idea generation workshop

1. Plan

The workshop design comprises three parts, the first, is a short introduction (15 minutes) to the thesis project, its context and objectives. Followed by information about the workshop itself and practical instructions to explain the process. The second and third parts of the workshop consist in the two following activities.

- Activity 1: Paper prototyping

A group activity (45 minutes) where participants are split into two sub-groups (3 + 3), each group was asked to collaboratively propose three distinct smart home scenarios. Scenarios should be represented visually using the provided tools, a list of these tools as well as their semantics are explained in the table below.

Tool	Description
Printed icons	Can represent one of the following entities: - Smart home connected device, examples from the VerisureIQ platform were illustrated: Smart plug, Climate sensor, smoke detector, Smart lock, Siren, Intrusion sensor. - Cloud data storage services: Xively - Custom devices: Arduino - Social media services: Twitter
Blank post-its	This is a blank template, can represent a device, an external service or any other object/entity which a smart home can interact with.
Blank paper sheets (A2 size)	Represent a canvas (or a screen) on top of which a scenario is expressed by placing icons/post-its and linking them to each other.
Marker pen	Used to draw lines on the canvas to represent connections between the entities.

Table 13: List of stationary used in workshop 1

Collecting data during this first activity has been done through observation (note taking) and taking pictures of the produced scenarios.

- Activity 2: Interview

This is an individual interview (10 minutes), where each participant is asked semi-structured questions. The following questions were prepared beforehand and answers have been written down on separate sheets of paper by the interviewer.

Question	Question

number	
1	Currently, when you prototype similar concepts, what methodology, workflow do you use?
2	Currently, when you prototype similar concepts, what tools do you use?
3	Do you think that visual programming is expressive enough?
4	Do you think having such prototyping environment will be useful for your work?
5	How do you intend to use it?
6	What other features or functionalities do you want to see in this prototyping environment?
7	Is there something you would suggest, that we could add in the next workshop?

Table 14: Questionnaire used in workshop 1

2. Printed icons used in the idea generation workshop

Smart Lock	Climate sensor	Smart Plug
Smoke detector	Siren	Intrusion sensor (lock unit)
twitter	ARDUINO	xively

Appendix II: Concept validation workshop

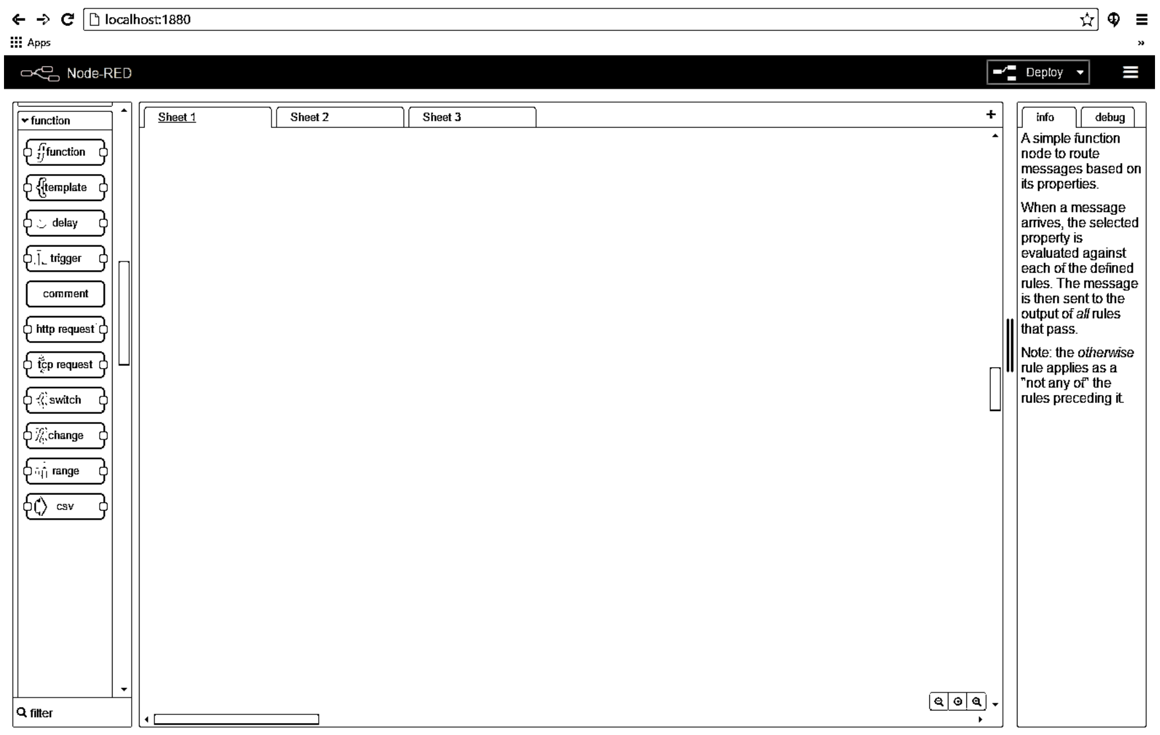
1. Plan

Following a short debriefing of workshop 1 and introducing aims and instructions for the current workshop, participants are split into 2 groups and are invited to engage in a paper prototyping exercise. The task is to visually program a set of predefined smart-home scenarios, one scenario at a time, in the form of message/control flow, hence using provided widgets as source, sink or combined source and sink elements.

We defined 2 scenarios and associated each of them to 2 different settings, each setting (or configuration) is aimed to test an assumption by including or excluding specific widgets (nodes).

The method to validate our proposed concept was inspired from user testing techniques, commonly used by UCD practitioners. In this workshop, we are interested in observing success or failure of participants to visually program each given pair of scenario/setting using the toolbox we provided. This metric is then enriched by information regarding extra elements (widgets) which reviled to be needed and eventually created by participants to achieve objectives in the exercise.

The list of stationary tools provided to participants is similar to the one used in our first workshop, however, we replace the blank sheet of paper, serving as canvas, with a wireframe version of the selected tool (node red).



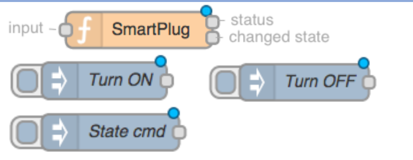
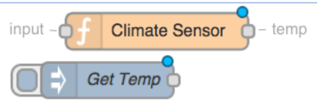
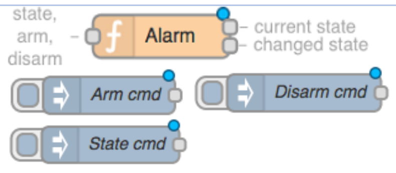
A simple function node to route messages based on its properties.

When a message arrives, the selected property is evaluated against each of the defined rules. The message is then sent to the output of *all* rules that pass.

Note: the *otherwise* rule applies as a "not any of" the rules preceding it.

Figure 21:Wire frame version of Node red designer

Printed icons in compliance with Node red look and feel are used in this work shop, we include a detailed list in the table below.

Widget name	Icon	Description
SmartPlug	 input SmartPlug status changed state Turn ON Turn OFF State cmd	Smart Plug device accepts 3 types of command messages (colored in blue) as input, and specifies 2 outputs. The first output returns current state (ON, OFF) upon request, and the second monitors state change and outputs the new value in real time.
ClimateSensor		Climate sensor device accepts 1 type of command messages (colored in blue) as input, and specifies 1 output. The output returns current temperature reading upon request.
Alarm	 state, arm, disarm Alarm current state changed state Arm cmd Disarm cmd State cmd	Alarm service (of a home installation) accepts 3 types of command messages (colored in blue) as input, and specifies 2 outputs. The first output returns current state (ARMED, DISARMED) upon request, and the second monitors state change and outputs the new value in real

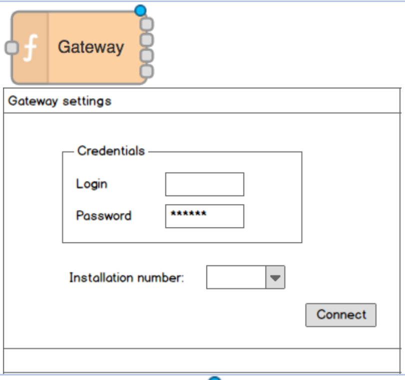
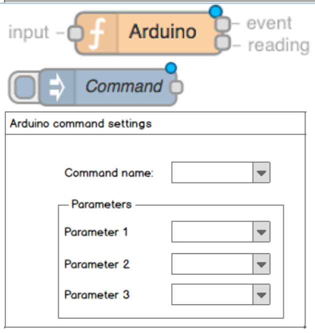
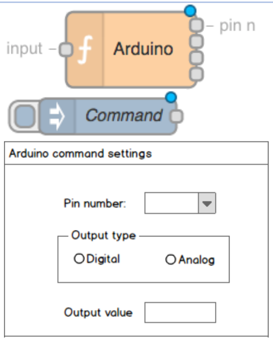
		time.
Gateway	 Gateway settings Credentials Login Password ****** Installation number: Connect	Gateway device physically represents a smart home installation, where all other devices are connected to it. When this widget is used, inputs of all other devices shall be wired to outputs of the gateway in order to explicitly represent the connection between them. The form shown underneath, is used to configure a given instance and bind it to one particular installation.
Arduino A	 Arduino command settings Command name: Parameters Parameter 1 Parameter 2 Parameter 3	Variation A of an Arduino based device for physical prototyping purposes, suggests a high level interaction pattern based on parametrized commands at input and specifies 2 outputs. The first outputs real time events pushed by the device internal logic, while the second returns current sensor readings upon request. The form shown underneath, is used to configure the command instance, it specifies the name and parameter values of the target command as implemented on device.
Arduino B	 Arduino command settings Pin number: Output type Digital Analog Output value	Variation B of an Arduino based device for physical prototyping purposes, suggests a low level interaction pattern based on writing/reading the state of physical pins. The widget receives at input, commands configured using the shown form, specifying details to override the value of a given pin. Outputs cover 4 pins, and periodically send out current state or sensor reading value if applicable.

Table 15: Printed icons used in concept validation workshop

2. Scenarios and settings used in the concept validation workshop

Scenario 1

Consider that we have 2 houses (HouseA and HouseB) connected to the smart home platform.
We want to implement the following logic:

If in HouseA the temperature reaches 26°
Then
Turn ON the lamp/smartPlug in HouseB
And
Turn OFF the oil-heater/smartPlug in HouseA

Scenario 2

Consider that we have in one house the following devices:

A climate sensor, and (a fan(motor) and a proximity sensor) wired to an Arduino device

We want to implement the following logic:

If the temperature reaches 25° C

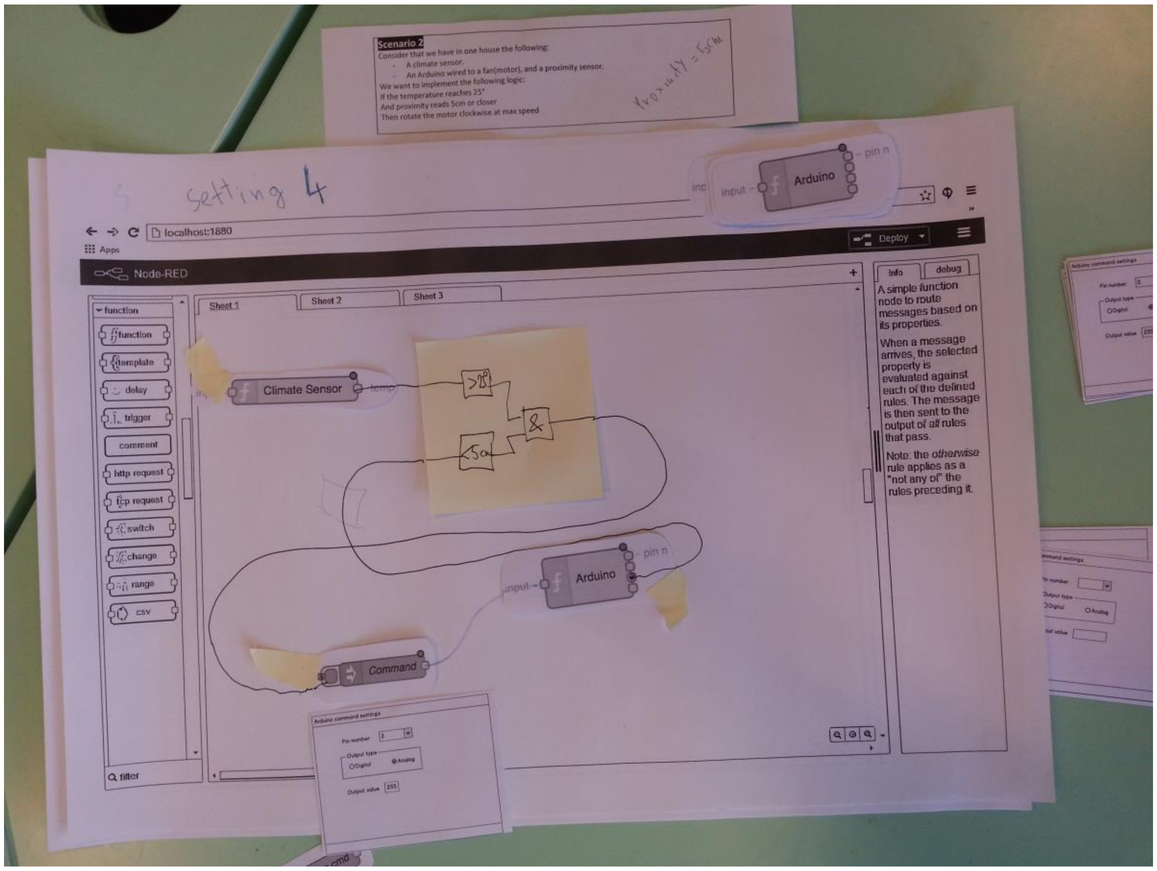
And proximity reads 5cm or closer

Then rotate the motor clockwise at maximum speed

	Included	Excluded	Comments
Setting 1	Gateway	Arduino A Arduino B	Scenario 1 is used to investigate preference of participants in regards to explicit versus implicit represenation of the Gateway device.
Setting 2	None	Gateway Arduino A Arduino B	
Setting 3	Arduino B	Gateway Arduino A	Scenario 2 is used to investigate preference of participants in regards to command level abstraction versus pin level abstraction for the Arduino device.
Setting 4	Arduino A	Gateway Arduino B	

Table 16: Details of settings used in the paper prototyping session

3. Photos from workshop


Scenario 2
Consider that we have in one house the following:
- A climate sensor.
- An Arduino wired to a fan(motor), and a proximity sensor.

We want to implement the following logic:
If the temperature reaches 25°
And proximity reads 5cm or closer
Then rotate the motor clockwise at max speed

A simple function node to route messages based on its properties.

When a message arrives, the selected property is evaluated against each of the defined rules. The message is then sent to the output of *all* rules that pass.

Note: the *otherwise* rule applies as a "not any of" the rules preceding it.

Arduino command settings
Pin number 2
Output type: Digital [✓ Analog]
Output value 255

Figure 19: Example of a visually programmed smart-home scenario

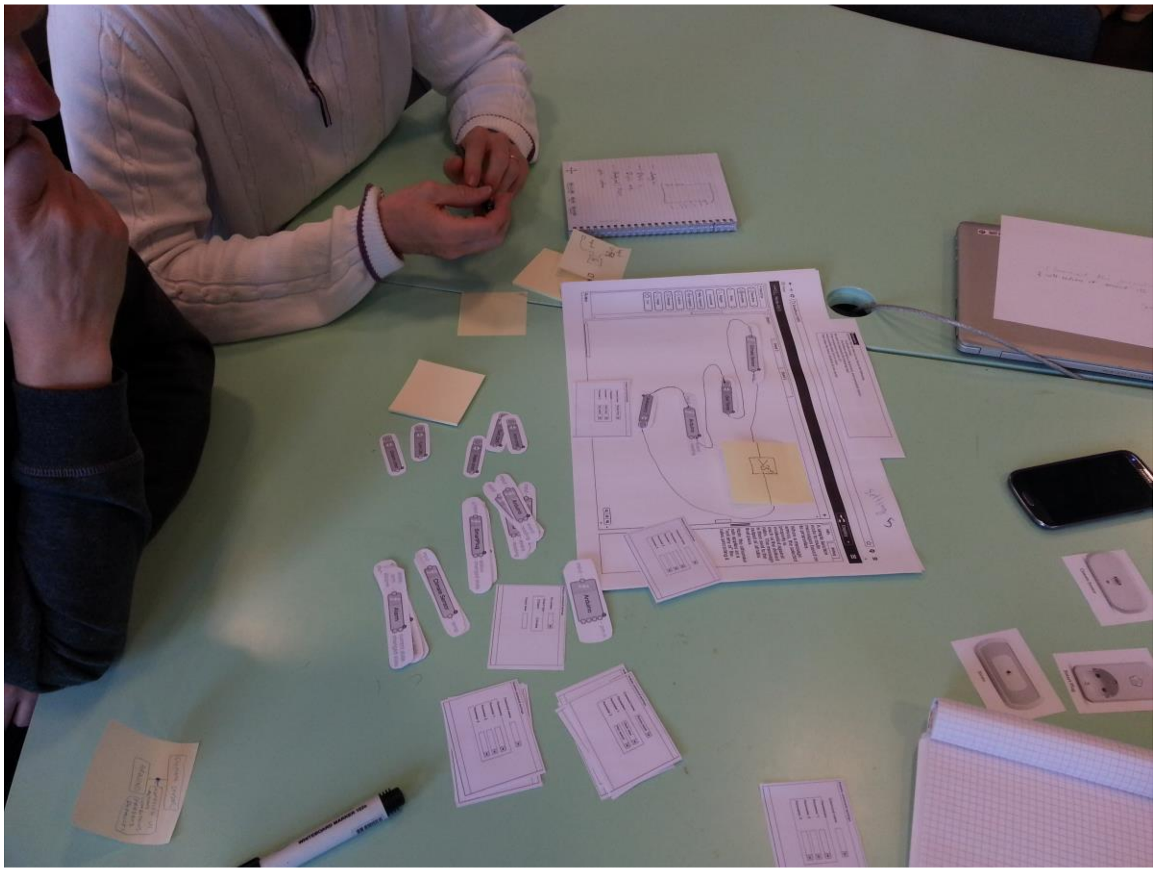


Figure 20: Participants visually programming a smart-home scenario

Appendix III: Evaluation workshop

1. Questionnaire used in the evaluation workshop

- Q1: Do you think that visual programming is expressive enough for rapidly prototyping smart-home concepts?
  Please give a score in the range [1-10] with 1 for: poorly expressive, and 10 for: highly expressive.
- Q2: Do you think having such prototyping environment will be useful for your work?
  Please give a score in the range [1-10] with 1 for: not useful, and 10 for: very useful.
- Q3: How do you intend to use it?
- Q4: do you think it's an extendable (or flexible) platform? How? any examples?
  Please give a score in the range [1-10] with 1 for: very rigid and limited, and 10 for: very flexible and easy to extend.
- Q5: what other features or functionalities are missing, or do you want to see in this prototyping environment?
  Please give a score in the range [1-10] with 1 for: not satisfied at all of the present feature set, and 10 for: the current feature set is very satisfactory.

The intention behind asking for numerical score input (quantitative) alongside semi-open questions (quantitative) is to detect any discrepancy between the two types of input, thus validating the obtained scores

8 References

[1] S. C. Mukhopadhyay, *Internet of Things : Challenges and Opportunities*. Dordrecht: Springer, 2014.

[2] L. Da Xu, S. Member, W. He, and S. Li, “Internet of Things in Industries : A Survey,” vol. 10, no. 4, pp. 2233–2243, 2014.

[3] K. Vredenburg, J.-Y. Mao, P. W. Smith, and T. Carey, “A survey of user-centered design practice,” *Proc. SIGCHI Conf. Hum. factors Comput. Syst. Chang. our world, Chang. ourselves - CHI ’02*, no. 1, p. 471, 2002.

[4] P. Taylor, G. Venturi, J. Troost, and T. Jokela, “International Journal of Human- People , Organizations , and Processes : An Inquiry into the Adoption of User- Centered Design in Industry People , Organizations , and Processes : An Inquiry into the Adoption of User-Centered Design in Industry,” no. November 2014, pp. 37–41, 2010.

[5] B. J. Mao, K. Vredenburg, P. W. Smith, and T. Carey, “User-centered design practice,” *Commun. ACM*, vol. 48, no. 3, pp. 105–109, 2005.

[6] M.-O. Pahl and G. Carle, “Taking smart space users into the development loop,” *Proc. 2013 ACM Conf. Pervasive ubiquitous Comput. Adjun. Publ. - UbiComp ’13 Adjun.*, pp. 793–800, 2013.

[7] E. Mainza, “A Visual Rapid Prototyping Environment for Smart Home Concepts,” 2017.

[8] V. Vaishnavi and B. Kuechler, “Design Science Research in Information Systems Overview of Design Science Research.” [Online]. Available: http://www.desrist.org/design-research-in-information-systems/.

[9] D. Koepsell, W.-P. Brinkman, and S. Pont, “Human Participants in Engineering Research: Notes from a Fledgling Ethics Committee.,” *Sci. Eng. Ethics*, vol. 21, no. 4, pp. 1033–48, Aug. 2015.

[10] S. EVELYNE, “FIFTY YEARS LATER: THE SIGNIFICANCE OF THE NUREMBERG CODE,” pp. 1436–1440, 1997.

[11] “WMA Declaration of Helsinki - Ethical Principles for Medical Research Involving Human Subjects.” [Online]. Available: http://www.wma.net/en/30publications/10policies/b3/index.html.

[12] D. R. Wright, “Research Ethics and Computer Science : An Unconsummated Marriage,” 2006.

[13] M. Langheinrich, A. Schmidt, and N. Davies, “A Practical Framework for Ethics – the PD-Net Approach to Supporting Ethics Compliance in Public Display Studies,” pp. 139–144, 2013.

[14] “EU research ethics policy.” [Online]. Available: https://ec.europa.eu/research/swafs/index.cfm?pg=policy&lib=ethics.

[15] *Ethics for researchers*. European Commission, 2013.

[16] M. J. Salvo, “Ethics of Engagement: User-Centered Design and Rhetorical Methodology,” *Tech. Commun. Q.*, vol. 10, no. 3, pp. 273–290, Jul. 2001.

[17] O. Mazhelis and P. Tyrv, “A Framework for Evaluating Internet-of-Things Platforms : Application Provider Viewpoint,” no. February 2011, pp. 147–152, 2014.

[18] C. Dixon, R. Mahajan, S. Agarwal, A. J. B. Bongshin, L. Stefan, and S. Paramvir, “An Operating System for the Home,” 2012.

[19] M. Jung, E. Hajdarevic, W. Kastner, and A. Jara, “Short paper: A scripting-free control logic editor for the Internet of Things,” *2014 IEEE World Forum Internet Things*, pp. 193–194, Mar. 2014.

[20] Y.-W. Kao and S.-M. Yuan, “User-configurable semantic home automation,” *Comput. Stand. Interfaces*, vol. 34, no. 1, pp. 171–188, Jan. 2012.

[21] M. Kovatsch, M. Lanter, and S. Duquennoy, "Actinium: A RESTful runtime container for scriptable Internet of Things applications," *2012 3rd IEEE Int. Conf. Internet Things*, pp. 135–142, Oct. 2012.

[22] M. Blackstock and R. Lea, "loT Mashups with the WoTKit," 2012.

[23] J. Chin, V. Callaghan, and A. Winckles, "Personalising the iCampus: An End-User Programming Approach," *2012 IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, pp. 347–351, Dec. 2012.

[24] B. E. Keiser, "Triggering Productivity," *Online Search.*, pp. 24–28, 2014.

[25] M. Rietzler, F. Schaub, J. Greim, B. Wiedersheim, M. Walch, and M. Weber, "homeBLOX : Introducing Process-Driven Home Automation," pp. 801–808, 2013.

[26] P. Sánchez, M. Jiménez, F. Rosique, B. Álvarez, and A. Iborra, "A framework for developing home automation systems: From requirements to code," *J. Syst. Softw.*, vol. 84, no. 6, pp. 1008–1021, Jun. 2011.

[27] W. K. Edwards, V. Bellotti, A. K. Dey, and M. W. Newman, "Stuck in the Middle : The Challenges of User-Centered Design and Evaluation for Infrastructure," no. 5, pp. 297–304, 2003.

[28] I. Sommerville, *Software engineering*, 9. ed., In. Addison-Wesley, 2011, p. 773.

[29] D. A. Norman, *The design of everyday things*. Basic Books, 2002, p. 257.

[30] H. Sharp, Y. Rogers, and J. Preece, *Interaction design: beyond human-computer interaction*, 2. ed., vol. 11. Wiley, 2007, p. 773.

[31] J. Simonsen and T. Robertson, *Routledge international handbook of participatory design*. Routledge, 2013, p. 294.

[32] K. Peffers, T. Tuunanen, C. E. Gengler, M. Rossi, W. Hui, V. Virtanen, and J. Bragge, "The design science research process: a model for producing and presenting information

systems research," *Proc. first Int. Conf. Des. Sci. Res. Inf. Syst. Technol. (DESRIST 2006)*, pp. 83–106, 2006.

[33] K. Peffers, T. Tuunanen, M. a. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.

[34] A. Hevner and S. Chatterjee, "Design Research in Information Systems," in *Design Research in Information Systems*, vol. 22, Boston, MA: Springer US, 2010, pp. 9–23.

[35] J. W. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*, 3rd ed., vol. 3rd. California: Sage, 2009, p. 260.

[36] R. a. Ribeiro, A. M. Moreira, P. van den Broek, and A. Pimentel, "Hybrid assessment method for software engineering decisions," *Decis. Support Syst.*, vol. 51, no. 1, pp. 208–219, Apr. 2011.

[37] T. Studt, "Are software wizards a precursor of intelligent analysis products?," *Res. Dev.*, vol. 39, no. 5, 1997.

[38] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," rfc 7252, 2014.

[39] "Lightweight Machine to Machine Technical Specification," OMA-TS-LightweightM2M-V1_0-20141126-C, 2014.

[40] "IPSO SmartObject Guideline: Smart Objects Starter Pack1.0," 2014.

[41] A. Ludovici and A. Calveras, "A proxy design to leverage the interconnection of CoAP Wireless Sensor Networks with Web applications.," *Sensors (Basel).*, vol. 15, no. 1, pp. 1217–44, Jan. 2015.

[42] A. Castellani, S. Loreto, A. Rahman, T. Fossati, and E. Dijk, "Best Practices for HTTP-CoAP Mapping Implementation," 07, 2013.

[43] B. Silverajan and T. Savolainen, “CoAP Communication with Alternative Transports,” 07, 2014.

[44] Z. Shelby, “Constrained RESTful Environments (CoRE) Link Format,” rfc 6690, 2012.

[45] C. Bormann and P. Hoffman, “Concise Binary Object Representation (CBOR),” rfc 7049, 2013.

[46] K. Li, R. Sun, and A. Rahman, “CBOR Equivalents of CoRE JSON Formats,” 2015.

[47] C. Alistair, “Walking skeleton,” 1996. [Online]. Available: http://alistair.cockburn.us/Walking+skeleton.