UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Andres Kiik

Business Process Decomposition & Distribution for Adaptive Internet of Things

Master's Thesis (30 ECTS)

Supervisor: Jakob Mass

Tartu 2018

Business Process Decomposition & Distribution for Adaptive Internet of Things

Abstract:

The Internet of Things (IoT) offers a great potential in many different application areas such as health care and home automation. However, in order to realize this potential, significant hurdles still have to be overcome. One of the hurdles is how to integrate IoT systems into business processes in the context of using standards such as BPMN2.0. In this thesis I present the results of a state of the art synthesis of Business Process Driven Internet of Things approaches. Based on the results, a business process modelling standard was chosen and used to overcome challenges which are related with IoT and home automation. The first challenge is the adaptiveness of the current IoT systems. For example, while new and more capable sensors enable complex applications such as smart door locks with face recognition that require significant computing resources, home controllers are often resource-constrained devices and therefore considered as a bottleneck for these complex systems. The second challenge is the interoperability of the current home automation systems because systems provided by different vendors cannot be controlled by the same application. To address these issues, this thesis proposes a software framework that enhances the adaptiveness and performance of IoT solutions. This is achieved through a novel approach where the process model is decomposed and the decomposed subparts are offloaded to an external entity. A prototype of this framework has been implemented using Camunda workflow engine. The framework supports IoT by integrating with the OpenHAB smart home system. The performance and scalability of the system is evaluated through a series of experimental case studies. Results showed that offloading can help in case of compute intensive tasks, a face recognition task performed three times faster for example.

Keywords:

IoT, Business Process, BPMN, Camunda, OpenHAB

CERCS: P160 Programming

Äriprotsesside dekompositsioon asjade interneti perspektiivist

Lühikokkuvõte

Asjade internet (IoT) pakub suurt potentsiaali mitmetes erinevates valdkondades. Selleks, et seda potentsiaali realiseerida, on vaja veel palju takistusi ületada. Üheks takistusteks on see, kuidas integreerida IoT süsteemid äriportsesside juhtimiseks mõeldud standarditega. Käesoleva magistritöö esimene osa annab ülevaate erinevate äriprotsesside standardite sobivusest IoT platvormiga. Teises osas võetakse luubi alla esimeses osas sobivaimaks tunnistatud äriprotsesside juhtimise modelleerimisstandard ja keskendutakse IoT platvormi probleemidele, mis on seotud koduautomaatikaga. Esimene väljakutse on selles, et kuna turule tuleb üha uusi IoT andureid ja seadmeid, mis võimaldavad suuremaid ja keerukamaid lahendusi nagu näiteks näotuvastusel toimiv tark ukselukk, siis need seadmed nõuavad ka suuremat arvutusvõimsust. Kodused kontrollerid on aga sageli piiratud ressursiga seadmed ning on seetõttu pudelikaelaks sellistes targa kodu süsteemides. Teine väljakutse on see, et erinevad tootjad tulevad turule enda koduautomaatika süsteemidega, mis omavahel suhelda ei suuda. Seega läheb vaja mitut rakendust, et tarka kodu juhtida. Nende probleemide lahendamiseks esitan kontseptuaalse IoT terviksüsteemi, mis võimaldab protsessid lõhkuda osadeks ja need osad käivitada teistes süsteemis, et vähendada koduse kontrolleri jõudlusvajadust. IoT seadmed selles terviksüsteemis on ühendatud kasutades platvormi, mis võimaldab erinevad koduautomaatika süsteemid üheks luua. Magistritöö raames valmis laiendus Camunda töövoo juhtimise platvormile, mis realiseerib esitatud terviksüsteemi. Esitan selle laienduse nõuded ja piirangud ning ilmestan neid kasutades targa kodu protsesse.

Võtmesõnad:

IoT, Äriprotsess, BPMN, Camunda, OpenHAB

CERCS: P160 Programmeerimine

Table of Contents

1 Introduction

1.1 Aim of the thesis

Business Process Management (BPM) and workflow technologies have proven to be an effective methodology for controlling processes, coordinating management and optimizing utilization of resources [1]. Using this potential in Internet of Things (IoT) may lead to higher efficiency and better resource utilization as well as would allow more expressiveness IoT automation rule engines and help to standardize IoT platform in terms of how to define the automation flows. As originally BPM was not designed for these new application areas such as IoT, the adaptation process faces with novel challenges [2].

One of the key application domains for IoT is home automation [3]. A typical home automation system contains of a home controller on which automation applications run and various sensors and actuators connect to. Home controllers are usually something that are installed once and are difficult or expensive to upgrade or replace. Therefore if new sensors come to the market which require more computing power, they cannot be installed to a current system because home controller is not capable for that. Even if a service provider provides a more powerful controller with an extra cost, it may be a waste of money because in real life, the resource requirements for various home automation applications can vary widely. For example, a smart door lock with passcode uses much less resources than a door lock with a camera and a face recognition.

While home automation systems and IoT overall are the application areas that keep growing [4], more and more IoT devices are available in the market. These devices are provided by different vendors and usually every vendor comes out with their own technology. Therefore, if there are devices at home bought from different vendors, it usually results in a poor user experience.

The IoT-aware business processes become more adaptive to the real world thanks to events that are detected by sensors and such events can occur at any time in the process. As affecting all possible activities, modelling such events into process is challenging and adds additional complexity.

This thesis aims to answer two research questions:

RQ1. *What kind of state of the art IoT solutions and approaches are there available from Business Process perspective?*

RQ2. *How can a business process model decomposed and deployed to an external entity?*

To address RQ1, a literature review is performed to find best approaches and solutions how a BPM can be used from the point of view of Internet of Things. Results show how different BPM standards meet the IoT platform needs.

To address RQ2, the architecture and implementation of a software framework for adaptive business process driven IoT is proposed. The approach covers business process model

decomposition and offloading with Camunda workflow engine, this helps to reduce the needed computing resources on a home controller. In addition to that, the system covers Camunda integration with OpenHAB, the latter one is an open source home automation software that connects different home automation systems and technologies into one single solution. The proof-of-concept implementation of the proposed approach is demonstrated and evaluated using two case studies, including a smart door lock scenario.

The rest of the thesis is organized as follows: Section 2 gives the reader a background of the key terms in this paper and contains a state-of-the-art literature review about the IoT-aware business processes. Section 3 presents the approach of the conceptual IoT system where the main part is the business process model decomposition. Section 4 evaluates the approach and results by means of a case study. Section 5 concludes this thesis and suggests future work.

2 State of the Art

This chapter provides background on key topics related to this thesis, including BPM, IoT, Camunda, OpenHAB, BPM decomposition, a study of the existing academic literature and overview of related works.

2.1 Background

2.1.1 Business Process Management

The following paragraph is based on the textbook by Dumas et al [1]. Business Process Management (BPM) is the art and science of overseeing how work is performed an organization to ensure consistent outcomes and to take advantage of improvement opportunities. It is the body of principles, methods and tools to design analyze and monitor business processes with the aim of improving their efficiency. A business process itself is a collection of activities, decisions and events involving a number of actors, objects and targeting to an outcome that is of value to a customer. BPM is not about improving an individual activity performance, it's about improving the whole business process. Depending on the objectives of the organization, the desired improvements may be different, but usually are related with the three dimensions of process performance: time, cost, quality.

As said, business processes are what companies do whenever they deliver a service or a product to customer. There can be several processes found in companies, two examples can be for example order-to-cash or issue-to-resolution processes. The first one starts when a customer submits an order to buy a product or a service and ends when it's deliver to the customer and the customer has made the corresponding payment. Issue-to-resolution process starts when a customer raises an issue, usually related to a defect in a product or service and ends when the issue has been resolved.

2.1.1.1 Components of a Business Process

A typical process involves a number of *events, activities, decision points, actors*, *objects* and *outcomes*. *Events* happen automatically and don't have a duration, for example time runs to 2 o'clock is an event. They can be used to trigger the execution of series of activities. An *activity* can be viewed as unit of work that takes time to complete. If an activity is rather simple and can be seen as single unit of work, we call it *task*. On the other hand, if the activity consists many steps, we consider it as *activity.* For example, we can consider sending an e-mail as a task, because it's straightforward, but validating client request as an activity if it contains multiple steps. Besides events and activities, a typical process involves *decision points*, which are points in time when a decision is made and that affects the way the process is executed. For example, the result of validation can be used in decision point to control what happens later in the process. In addition to these elements, a number of *actors* and *objects* can be involved with processes. Activities are performed by actors. Actors don't have be humans, software systems or organizations can be counted as actors as well. The process also involves immaterial objects, such as electronic documents, and physical objects, such as products. Last but not least, the

execution of a process leads to one or several *outcomes*. If the outcome delivers a value to the actors involved in the process, it corresponds to a *positive outcome*, if not, then *negative outcome.*

2.1.1.2 The BPM Lifecycle

Before the BPM can be applied, the organization needs to clarify what processes are they intending to improve. If an organization has engaged in BPM before, it is likely that business processes and their scopes are already defined, otherwise the identification part needs more resources. The lifecycle of BPM is shown in figure 2.1. This first step of a BPM initiative is termed as *process identification* and it leads *process architecture*, which typically takes the form of a collection of processes and links between these processes representing different types of relation.

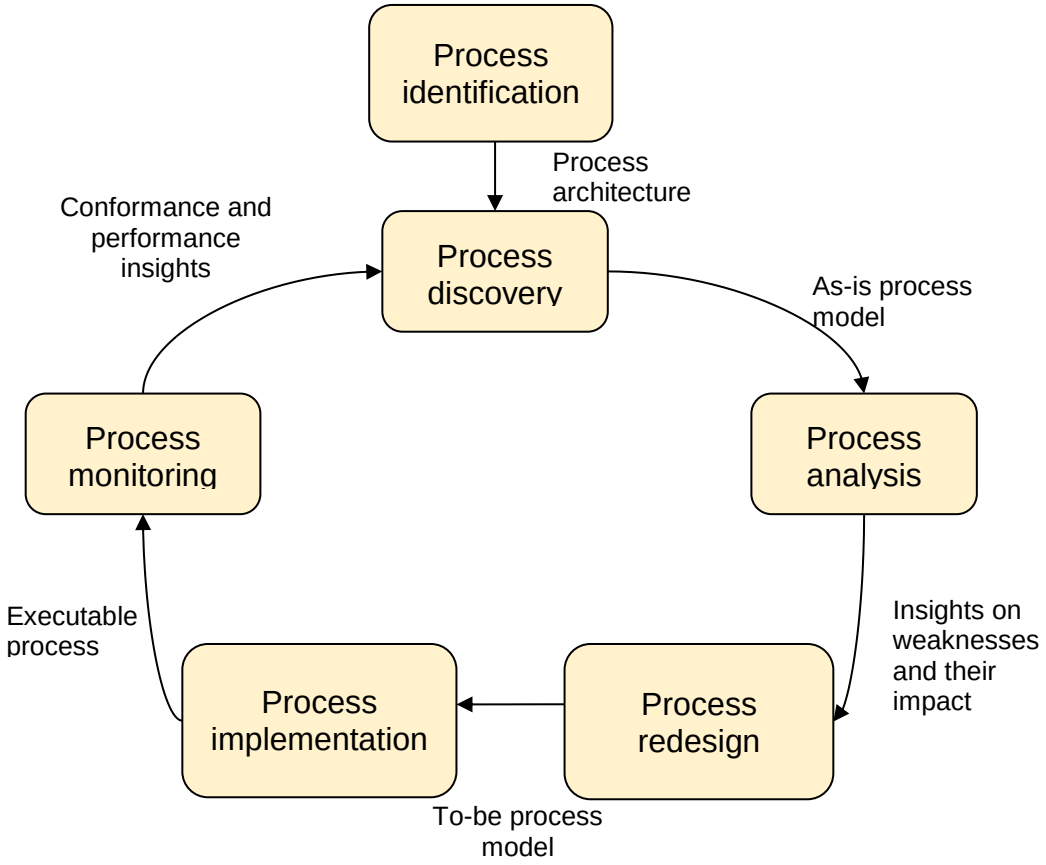

Figure 2.1. BPM lifecycle [1]

The next step in the lifecycle is to understand the business process in more detail, this phase is called *process discovery.* Typically, the goal of this task is to outcome one or several *as-is* process models that reflect how the work is done. These models have be easy to understand as

all the stakeholders involved in the BPM process must understand them. That is why these models usually are done diagrammatically, using different modelling languages. We will look in more detail one of the widely used standard in next section.

After having a deep understanding of as-is process model, the next phrase is to identify and analyze the issues in this processes, called the *process analysis* phase. In this step analysts need to obtain information that would allow them to find the main causes of the issues in the process.

When the issues have been analyzed, the next step is *process redesign*. In this phase analysts will consider multiple possible options for addressing these problems and can propose redesigned version of the process, called *to-be* process where the identified issues are addressed.

Once the process is (re)-designed, the next phase is *process implementation.* In this step, needed changes should be implemented so that the *to-be* process can be put into execution.

Finally, once the *to-be* process is running, relevant data needs to be collected and analyzed to determine how well the new process is performing with respect to its performance measures and performance objectives.

In this thesis, the (re)-design, execution and implementation phases of the lifecycle are more relevant than other parts.

2.1.1.3 Process Modelling with BPMN

BPMN stands for Process Model and Notation. It is a widely used standard for process modelling. The latest version of it is BPMN 2.0 [5] and it was released by the Object Management group in 2011.

There are four types of core elements provided by BPMN (figure 2.2). We recall that business process involves events, activities and decision points. Events stand for things that happen instantaneously while activities stand for units of work that have a duration. Also, it's important to keep in mind that events and activities are logically related in process. To show the sequence of the elements in the model, *sequence flows* are used. In BPMN, events are represented by circles, activities by rounded rectangles, decision points (called *gateways* in BPMN) are represented as diamonds and sequence flows are represented by arrows with a full arrow-head.

BPMN2.0 standard consists of the visual aspect and the serializable XML aspect but does say anything about execution. Using an engine software which interprets the XML and executes it, this can be achieved. Camunda is as an example of such an engine, described in the section 2.1.3.

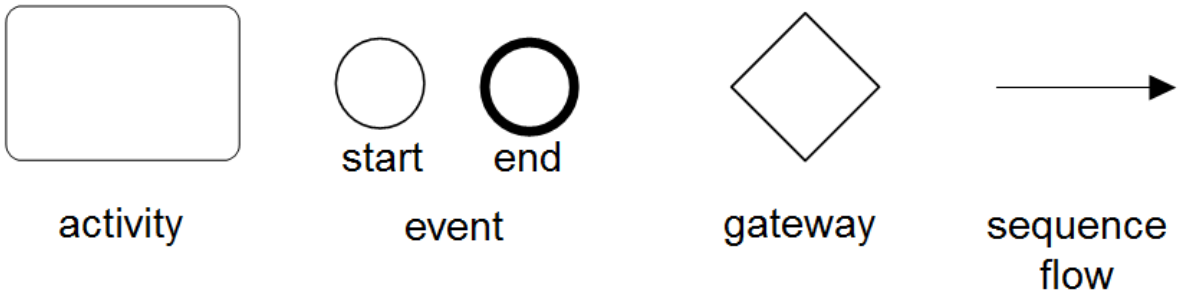

Figure 2.2. BPMN key elements [1]

2.1.2 Internet of Things

The IoT is a technology which helps to connect sensors, vehicles, hospitals, industries and consumers through internet connectivity [3]. It represents a comprehensive environment that consists of a large number of smart devices interconnecting heterogeneous physical objects to the Internet. The numerous smart devices enable smart environments in which the physical objects from our everyday life such as cars, homes, and appliances are connected.

Connectivity of these devices is the crucial pre-condition without which any of the other "smartness" cannot really take place. Things can be connected wired or wirelessly. In the Internet of Things wireless connection is the main way. Base on the existed infrastructure, there are many ways to connect a thing: RFID, ZigBee, WPAN, WSN, DSL, UMTS, GPRS, WiFi, WiMax, LAN, WAN, 3G, etc. Connected smart things makes interaction possible. [6]

2.1.2.1 Application Areas for the Internet of Things

There are many domains that have taken advantages of IoT as written by Gubbi et al [3], the popular ones are for example:

- **Building & home automation:** An extensive application area where IoT provides a wide range of new technologies for monitoring and controlling intelligent buildings and smart homes, by enhancing security to reduce energy and maintenance costs. With these control systems we make our home devices smart. Smart in the sense that, the devices can be monitored or perform task according to the defined instructions by user.
- **Smart cities:** Cost and resource consumption reduction can be achieved with IoT products for lighting, scrutiny, centralized and integrated system control. A smart traffic system in the city could reduce many road accidents. A good place for cost reduction is a smart traffic light which turns automatically on only if a vehicle approaches it. There are many ways to make a city 'smart'; the term city automation can also be used.
- **Health care:** IoT is also shaping modern technology used in health care and fitness industries by improving and enhancing the quality and accessibility of digital

products. IoT is making life more comfortable by bringing in many technologies in the field of healthcare. For example there are already a number of apps that track the heartbeat and help to monitor activity of the nervous system of human body. This system helps to take precaution before any health problem occurs.

- **Automotive:** The modern automobile can also benefit from IoT as it provides a wide range of cutting-edge automation from headlights to taillights including all the systems in between. Sensors in the car can sense the driver's soberness so in case the driver is drunk the car can automatically be stopped and therefore avoid possible road accidents.

2.1.3 Camunda

Camunda [7] is a light-weight workflow and Business Process Management Platform. Its core is a fast and solid BPMN 2 process engine for Java. It is open-source and distributed under the Apache license. Camunda runs in any Java application, on a server, on a cluster or in the cloud. It integrates with Spring[1] framework, it is lightweight and based on simple concepts.

Camunda mission is to make Business Process Management (BPM) ubiquitous by offering solutions that both business people and developers can use. There are plenty of BPM Suites that target solely business people. Camunda is unique because it takes developers seriously. The BPM engine needs to fit right inside any Java application architecture.

Camunda supports all aspects of Business Process Management (BPM) in the full context of software development. This includes non-technical aspects like analysis, modelling and optimizing business processes as well as technical aspects of creating software support for business processes. Camunda recognizes that BPM as a management discipline is a completely different aspect then BPM as software engineering.

There are numerous forms of BPM and workflow. Camunda's primary purpose and focus is to implement the general purpose process language BPMN 2.0.

2.1.3.1 Camunda components

The heart of Camunda project is Camunda Engine. It is a Java process engine that runs BPMN 2 processes natively. Besides the engine, Camunda provides modelling tools such as Camunda Modeler as well as web applications for management such as Camunda Admin and REST API [7].

[1] https://spring.io/

2.1.4 OpenHAB

Numerous smart devices and technologies arrive at our homes every day. Although, they are all aimed to enrich our lifestyle, they all lack one important feature: a common language they could speak to each other. This is when OpenHAB [8] comes into play by providing an integration platform to fix this issue.

OpenHAB is a software for integrating different home automation systems and technologies into one single solution that allows over-arching automation rules and that offers uniform user interfaces. OpenHAB is developed in Java, is vendor neutral and open source. It can run on any device that is capable of running Java Virtual Machine. It has different web-based UIs as well as UIs for iOS and Android.

OpenHAB does not try to replace existing solutions, but rather wants to enhance them - it can thus be considered as a system of systems. It therefore assumes that the sub-systems are setup and configured independently of OpenHAB as this is often a very specific and complex matter.

A core concept for OpenHAB is the notion of an "item". An item is a data-centric functional atomic building block - you can think of it as a "capability". OpenHAB does not care whether an item (e.g. a temperature value) is related to a physical device or some "virtual" source like a web service or a calculation result. All features offered by OpenHAB are using this "item" abstraction, which means that you will not find any reference to device specific things (like IP addresses, IDs etc.) in automation rules, UI definitions and so on. This makes it easy to replace one technology by another without doing any changes to rules and UIs.

Also, a very important aspect of OpenHAB's architecture is its modular design. It is very easy to add new features (like the integration with yet another system through a "binding") and you can add and remove such features at runtime.

2.1.4.1 OpenHAB REST API

OpenHAB provides a decent REST API, see table 2.1. It can be used to integrate OpenHAB with other systems as it allows read access to items and item states as well as status updates or the sending of commands for items. The REST API furthermore supports server-push, so there is a possibility to subscribe yourself on change notification for certain resources. Three media types are supported: XML, json and x-javascript.

Table 2.1. OpenHAB Item resources [8]

URI	HTTP Request Type
/rest/items	GET
/rest/items/{item.name}	GET, POST
/rest/items/{item.name}/state	GET

2.2 IoT-aware Business Processes

2.2.1 Review plan

I decided to use a systematic literature review research method thus the thesis was conducted based on the guidelines defined by Kitchenham and Charters [9]. To perform the process of the systematic review, the following activities were selected: definition of the research question, definition of the search strategy, the selection of studies, the extraction of data, and the implementation of a synthesis strategy.

2.2.2 Research Questions

RQ1: How do business process modelling standards fit with IoT integration?

RQ2: What are the main available IoT-aware business process solutions?

RQ3: What are the main challenges with IoT-aware business processes?

2.2.3 Search Strategy

In order to perform the search and selection of research studies, I formed the search terms, process itself and selection criteria.

2.2.3.1 Search Terms

I defined the following search strings:

C1: (“business”)

C2: (“process”)

C3: (“IoT” OR “Internet of Things”)

I found that the short term resulted more relevant papers that the long one. The final search string I used was: C1 AND C2 AND C3.

2.2.3.2 Search Process

I was using the Google’s Scholar web search engine to find the relevant papers for the study.

2.2.3.3 Study selection criteria

The selection criteria for selecting only the most appropriate studies defined as follows:

Inclusive criteria:

- The study must talk about how business processes can be integrated with IoT

Exclusive criteria:

- The study is not written in English
- The study is published before 2010

I decided to exclude the studies published before 2010 as I'm interested in latest technologies in this manner.

2.2.3.4 Data Extraction

The data extraction strategy is based on the data extraction forms that help to register the data obtained from the sources. [9]

The content of data collection form is following:

(a) Study title
(b) Author(s)
(c) Type of publication,
(d) Extraction date
(e) Year of publication
(f) Database

2.2.3.5 Synthesis Strategy

Table 2.2: Complete List of Selected Studies

Study ID	Study Title	Author(s)	Year
S1	Introducing Entity-Based Concepts to Business Process Modeling [10]	Sperner K., Meyer S., Magerkurth C.	2011
S2	A resource oriented integration architecture for the Internet of Things: A business process perspective [11]	Dar K., Taherkordi A., Baraki H., Eliassen F., Geihs K.	2015
S3	Internet Of Things-Aware Process Modeling: Integrating IoT Devices as Business Process Resources [12]	Meyer S., Ruppen A., Magerkurth C.	2013
S4	IoT-aware business processes for logistics: limitations of current approaches [13]	Ferreira P., Martinho R., Domingos D.	2010
S5	The Real-time Enterprise: IoT-enabled Business Processes [14]	Haller S., Magerkurth C.	2011
S6	Towards modeling real-world aware business processes [15]	Meyer S., Sperner K., Magerkurth C., Pasquier J.	2011

The search for the review was performed in October 30th, 2016. It resulted about 32, 400 according to Google's Scholar. After investigating the results, the first three pages of the search result seemed somewhat relevant, so I decided to take into to the second round 30 papers resulted by Scholar. After applying the defined search criteria to these papers, only 6 of them were selected. The list of selected studies is shown in table 2.2.

2.2.4 Review results

Table 2.3: List of studies classified by research questions

Research question	Studies(s)
RQ1: How does business process modelling standards fit with IoT integration?	S1, S3, S4, S6
RQ2: What are the main available IoT-aware business process solutions?	S2, S4
RQ:3 What are the main challenges with IoT-aware business processes?	S4, S5

The results are divided into sections, each section is trying to find answer one of the defined research question. As I assumed, since the topic is quite novel and often related as open research challenge, there weren't many relevant studies which the results could be based on. In table 2.3, I will list the classification of studies based on how they contribute to answer the defined research questions.

2.2.4.1 How do business process modelling standards fit with IoT integration?

S3 and S6 clearly state that besides the most commonly used business process modelling standards such as Web Service Business Process Execution Language(WS-BPEL), Event-driven Process Chain(EPC) and Unified Modelling Language (UML), BPMN 2.0 is the best IoT-aware state-of-the-art process modelling approach. S6 explains that so far the latest versions of BPMN, EPC and UML only provide graphical flow-chart process notations.

Therefore, several metamodels have been developed to convert these notations into executable ones such as WS-BPEL 2.0. The latter one is the leading standard for representing executable business processes. WS-BPEL is an XML based process representation that allows for composing web services to higher value services. It is processed in a block oriented way and therefore heavily differs from graph based notations. Due to the totally different concepts, conversions from BPMN 1.2 or EPC to WS-BPEL are still facing with many problems. S6 concludes that with the inclusion of defined execution semantics and an XML serialization format BPMN 2.0 is the first and only notation combining an end-user friendly notation with a detailed technical specification of an executable model in the same representation.

In addition to S3 and S6, S1 uses the BPMN 2.0 as a base standard to bring together the IoT and Business Process Management.

On the other hand, WS-BPEL is chosen as a business process modelling standard in S4 to implement IoT-aware business processes for logistics. Since there is no word why this standard was chosen, it's difficult to say if it's the best standard to achieve the goal. Nevertheless, the study identifies the main limitations in the BPEL, regarding the support of design and runtime in supply chain management processes with business processes.

WS-BPEL limitations according to S4:

- In design time:
  - WS-BPEL fails in allowing for a compact process definition for a larger number of exceptions and alternative paths, which cannot be foreseen or practically defined. WS-BPEL supports only the handling of predicated exceptions with the exception handlers as well as alternative flows through the use of if/else blocks.
  - All process perspectives must be defined in a static way and beforehand, meaning that process definitions cannot be incomplete or dynamically specified.
  - The definition of business logic that is to be run in smart items is not possible with WS-BPEL. It would be valuable to access and specify all sub-process definitions that compose a business process model together.
  - WS-BPEL does not foresee the distribution of business logic between a central system and smart items.
  - WS-BPEL does not allow controlling which, how and by whom parts of a process definition can be changed.
- In runtime:
  - The lack of support for changes of business process instances, because of unpredicted, ad-hoc circumstances
  - The lack of support in migrating instances from old process definitions to new ones, if a redefinition of the business process occurs.

Even though BPMN 2.0 is the best IoT-aware modelling standard that provides coverage for more IoT specific properties than other approaches, there still are some challenges and BPMN

cannot be used as an out-of-the standard for IoT integration. S1, S3 and S6 propose an extension to current standard to address these limitations and challenges. S1 and S3 takes the IoT domain model as basis to describe business process view of this model and to find the mapping between IoT concepts and BPMN concepts and thereby providing an extension to the BPMN 2.0 standard where the challenges are addressed.

S1 takes the first version of the domain model for the IoT from the IoT-A project [16] to find the mapping of IoT concepts to current BPMN concepts and their sufficiency for modeling of IoT aware process. Results are presented in table 2.4.

Table 2.4. Mapping between IoT concepts and BPMN concepts [10]

IoT concept	BPMN concept	Sufficiency for Modeling
Physical Entity	TextAnnotation	not sufficient
Virtual Entity	DataObject	sufficient
Augmented Entity	-	not needed in BPMN
Sensor	Participant	sufficient
monitoring	ServiceTask	not sufficient
Actuator	Participant	sufficient
acting	ServiceTask	not sufficient
Tag	-	not needed in BPMN
Resource	-	not needed in BPMN
Service	ServiceTask	not sufficient
User	Participant, Event	sufficient

2.2.4.2 What are the main available IoT-aware business process solutions?

S2 states that Service Oriented Architecture (SOA) or the Resource Oriented Architecture (ROA) are normally used for service-based IoT services. S3 and S4 mention SOA as preferred approach regarding interactions of information system with more powerful smart items.

The SOA-based approaches provide a uniform and structured communication with low power IoT devices as in e.g. TinySOA [16], compression of the Simple Object Access Protocol

(SOAP) messages for sensor devices, proprietary communication protocols, implementation of WS eventing and compiling and deploying the BPEL code up to the edge of the network. [S2]

The ROA-based techniques exploit the Representational State Transfer (REST) design principles to create, for instance, Web of Things, or simple mash-up applications. [S2]

S2 also claims that up to now studies about above mentioned approaches are still lacking a unified design for end-to-end integration of both enterprise level business processes and tiny IoT devices. This especially applies for ROA based approaches.

S2 proposes a ROA based architecture of the design and implementation of a generic architectural model which provides core components required for an end-to-end integration of IoT systems with special focus on IoT-aware BP-s. The proposed framework contains necessary APIs to invoke IoT services which ease the life of IoT-aware BP developers, an event-based integration model built upon a well-known publish-subscribe mechanism, a dynamic service replacement facility upon failure of IoT devices and decentralized execution of the BP. Besides the proposition, the study includes implementation and evaluation of the framework, including installing BPMN execution engine to Android.

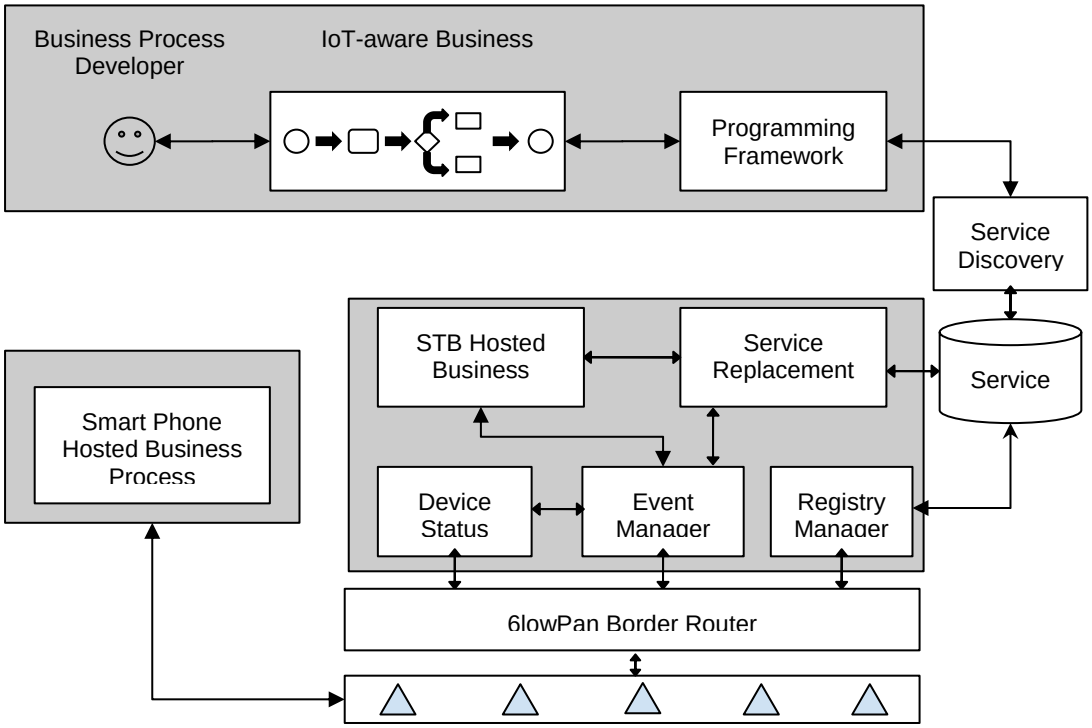

Figure 2.3. The proposed ROA framework by S2 [11].

2.2.4.3 What are the main challenges with IoT-aware business processes?

All of the reviewed studies stated that when it comes to integrating IoT aspects into business process modelling, there are still more or less challenges to overcome.

S5 sums up most of the challenges that IoT-aware processes face:

- *Adaptive and event driven processes:* The IoT-aware business processes become more adaptive to the real world thanks to events that are either detected directly or by real-time analysis of sensor data and such events can occur at any time in the process. As affecting all possible activities, modelling such events into process is cumbersome, adds additional complexity and makes the processes difficult to understand. Moreover, how to react on a single event can depend on the context.
- *Processes dealing with unreliable data:* A degree of unreliability and uncertainty is introduced into the processes when the data is coming from the physical world via smart items, e.g. sensors. This raises the need for the quality of information (QoI) so that the process modeller is able to define thresholds that reflects the degree of certainty. Things get more complex if multiple events are involved. From a BPM perspective, the challenge is how to capture, process and express such information in the used modelling notation language.
- *Processes dealing with unreliable resources:* In addition to the unreliable data that comes from resources, the resources themselves have some uncertainty attached. Processes relying on such resources need to be able to handle such situations. It is challenging because when we're talking about resources that might generate an event at one point in time, not receiving any event can be due to resource failure, but also because there was nothing to report.
- *Highly distributed processes:* The decomposition and decentralization of business process can make sense when interacting with real world objects and devices. The challenge is how to define the process centrally, including the fact that some activities will be done remotely. It should be possible to deploy the related services to smart items and then run and monitor the complete process.

2.2.5 Discussion

With this systematic literature review about IoT-aware business processes I presented state of the art analysis of available business modelling standards for IoT integration, the architectural approaches and the main challenges that IoT-aware business processes face. From the analysis of this review I observed that:

- BPMN 2.0 is the best IoT-aware modelling standard that provides coverage for more IoT specific properties than other most commonly used business process modelling standards such as WS-BPEL,EPC and UML.
- Service Oriented Architecture and Resource Oriented Architecture are normally used as a reference architecture for IoT-aware business process integration.

- The main challenges with IoT-aware business processes are how to make: (a) adaptive and event driven processes (b) processes dealing with unreliable data (c) processes dealing with unreliable resources (d) highly distributed processes

The review gives a great overview what is the current situation when it comes to integrating BPM aspects into IoT and what are still open research challenges.

2.3 Related works in Business Process Model decomposition

One of the BPM challenges is how to decompose BPM models so that parts of a business processes can be deployed to and executed on external entity. For example, let us take a business process with several activities but only some of the activities are critical for the company because of the fear of losing or exposing sensitive data. With decomposition these critical tasks could be transformed to a secure environment. Meanwhile the less important parts of the model can be executed on a third party environment, e.g. on a cloud service. Cloud computing gives the opportunity to use computing resources in a pay-per-use manner and perceive these resources as unlimited. Also, a decomposition can be used for home automation systems, for example, to hand out the processing power.

This section presents existing works which are directly similar to the goals of RQ2 of this thesis.

2.3.1 An Extensible Home Automation Architecture based on Cloud Offloading

To address limitations that current home security and automation systems usually use a local home controller which is not capable of handling demanding computing resources, authors in [18] presents a new Cloud-Enhanced Home Controller (CEHC) architecture where the resources of the local home controller are augmented with external cloud resources accessed over the network, shown in figure 2.4.

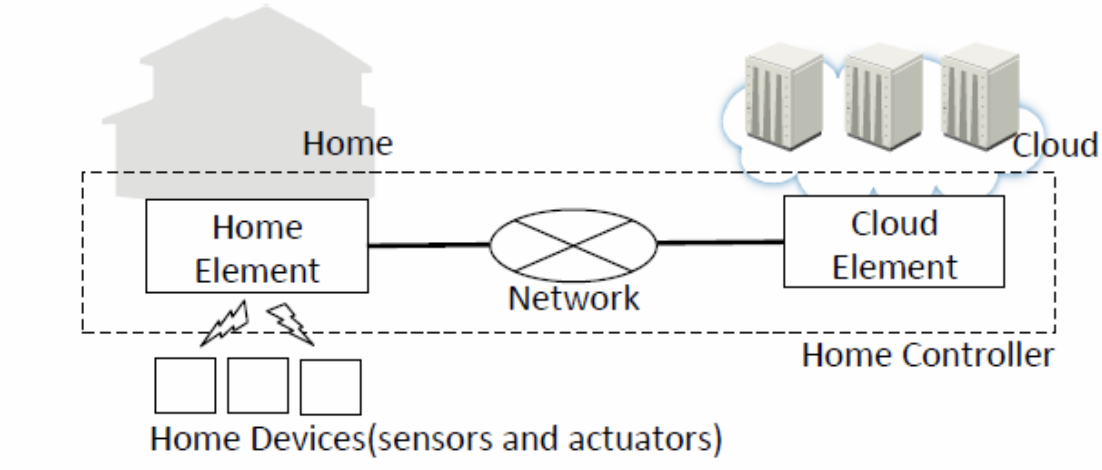

Figure 2.4. Cloud Enhanced Home Controller [18]

With this approach, applications run on the local controller box when they can, and then partially or fully migrate to a cloud virtual machine as and when needed. This allows home controllers to be provisioned more conservatively, while providing a built-in ability to cope with new applications as they emerge. First contribution of this paper is to propose such a cloud-enhanced home controller architecture. This includes supporting platform services, and a novel application design paradigm where applications are designed to operate in three possible modes: completely at the home controller, mostly in the cloud, and in a degraded mode when the cloud or network is unavailable. The second contribution is a system-wide scheduler (SWiS) that determines the optimal placement for a given set of applications on the home controller and in the cloud. Finally, they validate the architecture using a number of realistic home control applications constructed for this platform, one of them is shown in figure 2.5. The results demonstrate that offloading reduce the computing load of a home controller.

CEHC design

There are three operation modes that an application can have: Home Mode is when all the application functions run on the Home Element (HE), Cloud Mode is when some functions run on the HE and other on the Cloud Element (CE), Degraded Mode is when there is a reduced-functionality version of the application that runs on the HE. This mode could be used in case of an application is executing in Cloud Mode but the network is disconnected. Also, this can be used when HE is overloaded to accommodate a full Home Mode of an application.

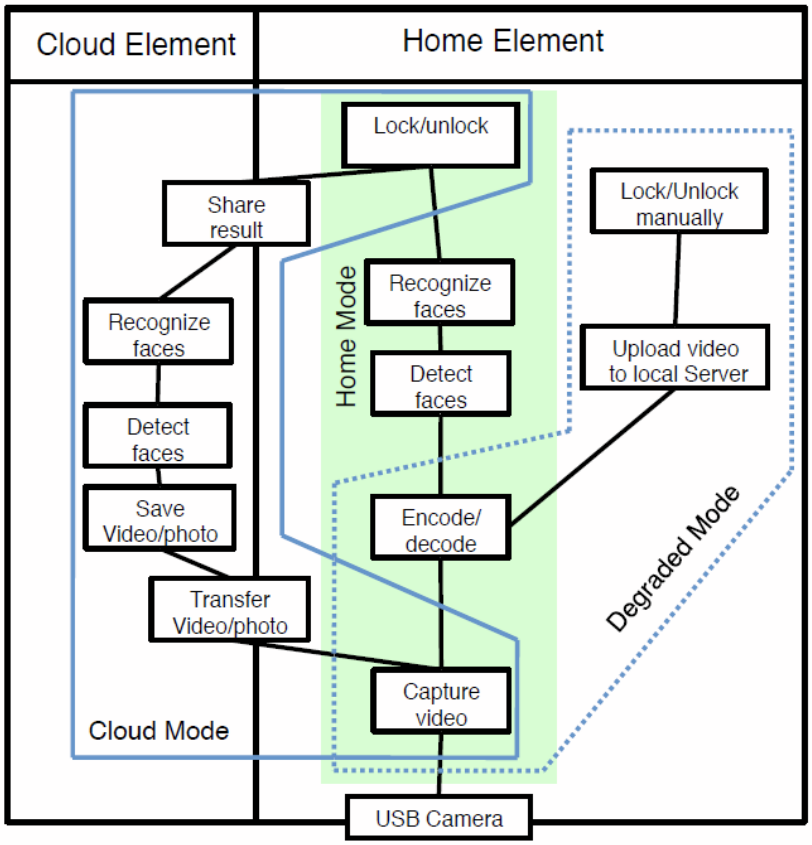

Figure 2.5. An example application: Door lock application [18]

System-wide scheduler (SWiS)

The SWiS scheduler decides which modes all applications installed on the CEHC should have based on factors such as the CPU load of the HE, the memory consumption of the HE, the CPU load of the CE, the memory consumption of the CE, network bandwidth, and the cost of the CE. To determine the operating modes of each of the applications, the SWiS scheduler needs to know about the resource consumption of the different application modes. This paper assumes that the compute resources and network bandwidth required by all modes of an application are known before installation. SWiS supports dynamic updating since it monitors resources such as CPU load, memory consumption and network bandwidth and therefore can change the selected pattern as conditions evolve. For example, when a network connection breaks, SWiS considers all patterns that use the CE as infeasible and leads automatically to the best solution using only local resources on the HE.

2.3.2 Towards Situation-Aware Adaptive Workflows

Authors in [19] presents an approach that allows workflows to become situation-aware to automatically adapt their behaviour according to the situation they are in. They concentrate on the derivation of higher level situation information from lower-level context and sensor information to adapt situation-aware workflows based on the recognized situational changes.

The paper introduces (i) a concept and an architecture for a situation-aware workflow management system called SitOPT, (ii) a new workflow concept for modelling situation control flows, (iii) situation templates (ST) to support the modelling of the situation-recognition process and (iv) a common situation model for storage and handling of the detected situations.

2.3.3 UAV-Based IoT Platform: A Crowd Surveillance Use Case

The article [20] demonstrates how drones can be used for crowd surveillance based on face recognition. The authors study the offloading of video data processing to a mobile edge computing (MEC) node since such use case requires much computational overhead. For face recognition they use the Local Binary Pattern Histogram (LBPH). They compared the performance process against the local processing of the face recognition on board of drones. The result of this experiment demonstrates that it is highly efficient to offload the face recognition operation to MEC rather than processing it locally on board.

2.3.4 Discussion

[18-20] propose an approach of an adaptive workflow-systems, [18,20] demonstrate how offloading can help to reduce the computing load on a local/home controller while [19] is focused only on situation-aware workflows. Although, the goals about adaptiveness meet with my thesis, none of the papers address the challenges using the BPM advantages.

3 Approach

Most of the current home automation systems run on a local controller at home. However, the local controller may become a resource bottleneck. The second subject matter is that usually every vendor provides their own technology for their IoT automation system or device which means that multiple applications are needed to control a smart home. To address these issues, a new approach is proposed, the architecture of this is shown in diagram 3.1. The approach for how IoT-aware business processes can be used to control a home automation system using Camunda workflow engine with OpenHAB platform is covered in section 3.2. The most of the effort goes into introducing the approach for business process model decomposition and offloading using Camunda workflow engine, this is covered in section 3.3.

3.1 Architecture

The central component is the *master*. The *master* can run BP-s, additionally it supports IoT features in the BPs through OpenHAB. The *master* also has the capability of breaking processes into subparts and offloading these subparts to other nodes called *workers*. The architecture supports multiple *workers*.

The connectivity between nodes is based on HTTP + REST. The *master* is based on Camunda engine + a special plugin, worker is just a Camunda engine, IoT platform is based on OpenHAB.

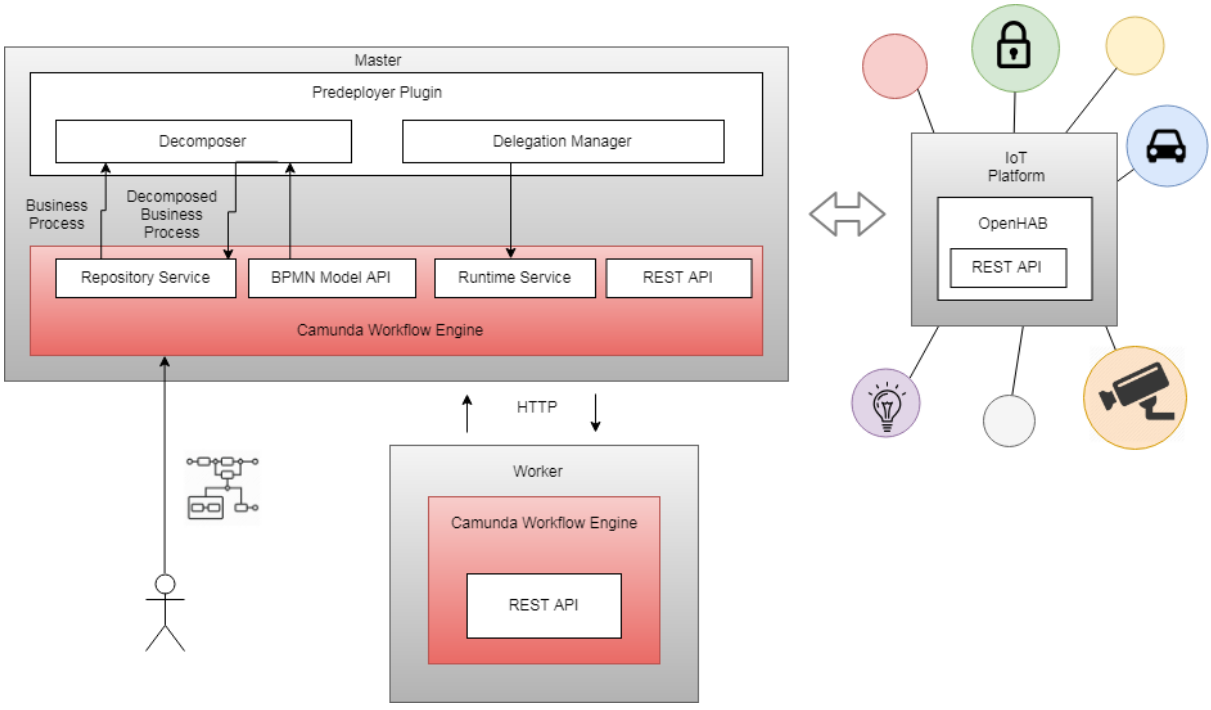

Diagram 3.1 Overall architecture of the system

3.2 Camunda Integration with OpenHAB

In order to integrate the *master* node with the OpenHAB IoT platform, a Java application was created. It uses Camunda framework to create and execute IoT-aware business process models and it also connects to OpenHAB RESTful API. To achive the goal, Eclipse Mars, Java Maven project with Camunda platform and Camunda Modeler was used. One of the goals was to keep the application as lightweight as possible, because it should be runnable on Raspberry Pi. The latter requirement is apparent if we are talking about smart home systems which should be easy to integrate and low-costed.

After being familiar with Camunda and OpenHAB, it was concluded that Camunda ServiceTask is the process task type that could be used for making the HTTP request against OpenHAB RESTful API. ServiceTask is used to invoke an external Java class and is visualized as a rounded rectangle with a small gear icon in the top-left corner, see figure 3.3. XML representation of the ServiceTask where the class is specified looks as following:

```
<serviceTask id="javaService" name="My Java Service Task"
activiti:class="org.activiti.MyJavaDelegate" />
```

Moreover, there is a possibility to inject values into the fields of the delegated classes. To inject values that are dynamically resolved at runtime, expressions can be used.

To implement a class that can be called during process execution, this class needs to implement the *org.activiti.engine.delegate.JavaDelegate* interface and provide the required logic in the execute method.

To integrate Camunda with OpenHAB, OpenHABHttpServiceTask class was implemented and this class can be binded with OpenHAB related ServiceTasks and which is responsible for performing HTTP requests to OpenHAB RESTful API. This is shown in figure 3.2.

```
// Create Camunda process engine
ProcessEngine                              processEngine                              =
ProcessEngineConfiguration.createStandaloneInMemProcessEngineConfiguration()
.setDatabaseSchemaUpdate(ProcessEngineConfiguration.DB_SCHEMA_UPDATE_FALSE)
.setJdbcUrl("jdbc:h2:tcp://localhost/~/test")
.setAsyncExecutorEnabled(true)
.setAsyncExecutorActivate(true)
.buildProcessEngine();
        processEngine.getProcessEngineConfiguration().getAsyncExecutor()
.setDefaultTimerJobAcquireWaitTimeInMillis(1);
// Get Camunda services
RepositoryService repositoryService = processEngine.getRepositoryService();
RuntimeService runtimeService = processEngine.getRuntimeService();
// Deploy the process definition
repositoryService.createDeployment()
      .addClasspathResource("MyServiceTask.bpmn")
      .deploy();
```

Figure 3.1 Camunda process engine configuration

```
public class OpenHABHttpServiceTask implements JavaDelegate {
       private Expression requestUrl;
       private Expression requestType;
       private Expression requestData;
       private Expression resultVariableName;

private enum HTTP {
    GET, PUT, POST
}

public void execute(DelegateExecution execution) {
String type = ((String)requestType.getValue(execution));
String url = ((String)requestUrl.getValue(execution));
switch (HTTP.valueOf(type)) {
case GET: performHttpGet(url, execution);
               break;
case PUT: performHttpPut(url, execution);
               break;
case POST: performHttpPost(url, execution);
               break;
}
       …
```

Figure 3.2 OpenHABHttpServiceTask class

In the main method as an entry point of the application, the Camunda process engine is configured and the process definition is deployed, see figure 3.1. H2 database was used.

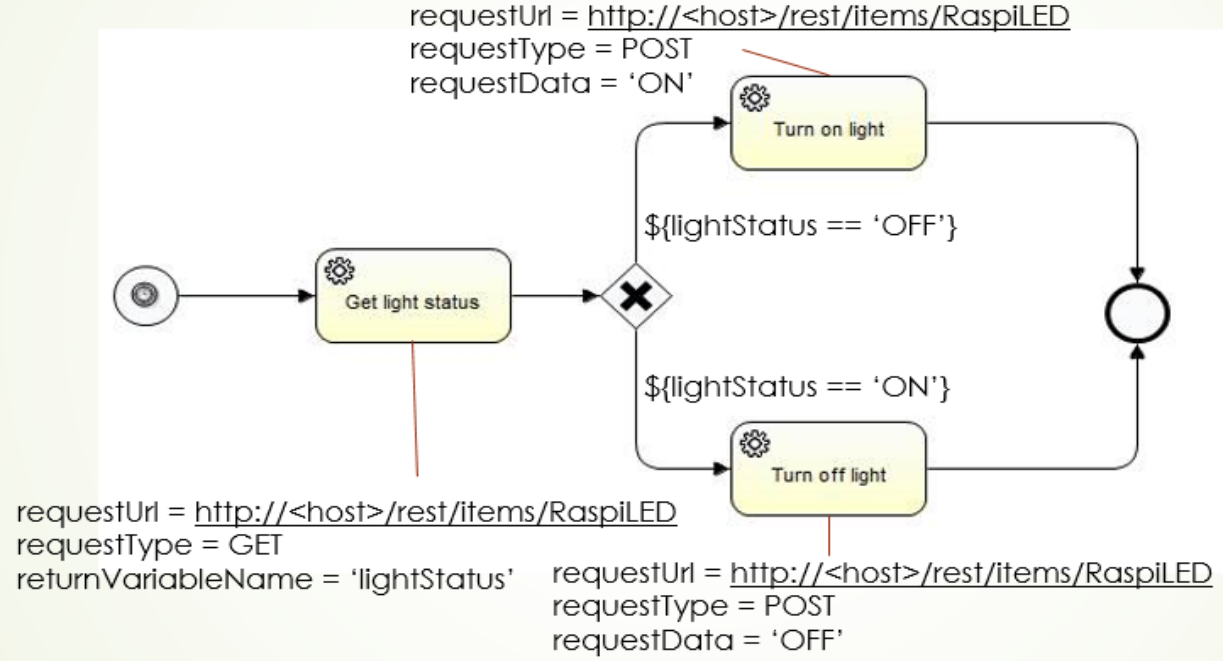

Figure 3.3. BPMN2.0 model for blinking the LED light

A very simple BPMN2.0 model was defined to demonstrate the integration, shown in figure 3.3. The basic idea of what the whole thing was supposed to do was to blink the LED after every second. The model starts with start timer event that is triggered after every second, the first activity is responsible for getting the light status from OpenHAB with GET request, then comes the XOR gateway where the light status is checked and according to the status, the light is turned on or off with POST request.
Raspberry B+ with wires and LED lights were provided to set up my environment. The result of the engineering is shown in figure 3.4.

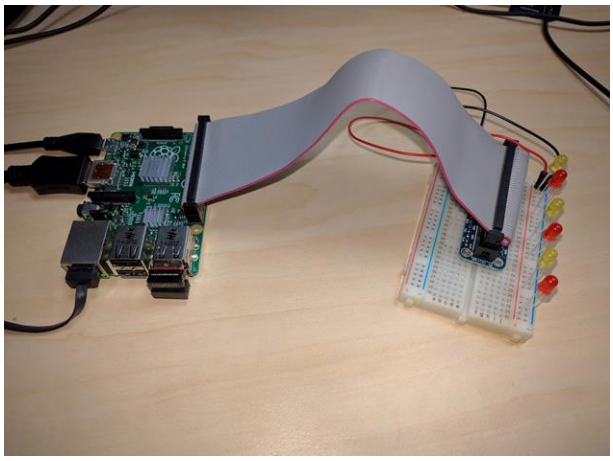

Figure 3.4. Engineering for the Blinky LED

The implemented application ran on HP EliteBook 840 laptop, OpenHAB on Raspberry Pi, they were connected with an Ethernet cable to each other.

Since the implemented OpenHABHttpServiceTask class is generic, and that thus using it you can access various items from openHAB by simply adjusting the parameters in the process definition, no extra Java programming necessary.

The OpenHAB configuration setup on Raspberry Pi is based on instructions made by Erich Styger [21].

3.3 Process decomposition and offloading

To address the issue when the *master* node has limited capabilities, the proposed framework enables the *master* node to offload tasks to the *worker* node. Before tasks can be offloaded, the process must be broken into subparts, this is called a decomposition. In this thesis we assume that the environments where the processes run and are deployed use the same business process engines which is responsible for running the process. We also assume that the activities which should be assigned to external entity have been marked as offloadable by the process designer. There is also one assumption how the processes are defined: if there are two or more activities in a process in a row that should be deployed, these activities should be moved to a sub-process and this sub-process itself is assigned to external entity. The last assumption helps to optimize the process decomposition.

In contrast of the transformation based approach introduced in [22] where an intermediate model was defined and used, we try to allow the automated transformation without defining an additional modelling language and therefore get rid of extra transformations that are needed for converting a business model to the intermediate model and back. We try to achieve using the metamodel elements of the business process model language as was the initial plan in [22] but as they tried to find a solution in more general manner and support several business process languages it got too complicated and difficult to manage so they abandoned this approach. We are focused on one process modelling language in this paper, BPMN2.0, and our goal is to use as much elements as possible from this standard and stay using so-called vanilla BPMN2.0.

Figure 3.5. An example business process

In figure 3.5, an example of a business process is shown using BPMN2.0 modelling language. It is a very simple process with three activities, a start and an end event. The highlighted activity, Activity 2, is the one that we want to deploy to a *worker*. For simplicity, it is a single activity in this case, but it can be a sub-process as well and contain more activities or other model elements.

Now, there are basically two executors: the *master* where the main process runs and the *worker* where the activities marked as external should run.

As single activity or sub-process itself is not a complete process model, since a correct process model has a start and end event, there is a need for a new process model where the missing ends are added as shown in figure 3.6.

Figure 3.6. A complete process model for deployment

To make these processes to communicate with each other, we need additional activities or events for that. In [22] four possible communication nodes are introduced: invoke-request, invoke-response, receive and reply.

The invoke-request node is used for invoking a process. It has one outgoing edge which points to a receive node, located in the process that is invoked. The invoke-response node is used as synchronization node for communication with other processes. It waits for a response from the other process before continuing its execution. It has one incoming edge which originates from a reply node, located in the process that is invoked. Using these nodes we can model a new process, but we need to consider both synchronous and asynchronous communication cases here.

First, in the case of synchronous communication, the main process uses an invoke-request node to invoke the other process and then proceeds to invoke-response which waits until the other process finishes its execution, before continuing with the process. This is shown in figure 3.7.

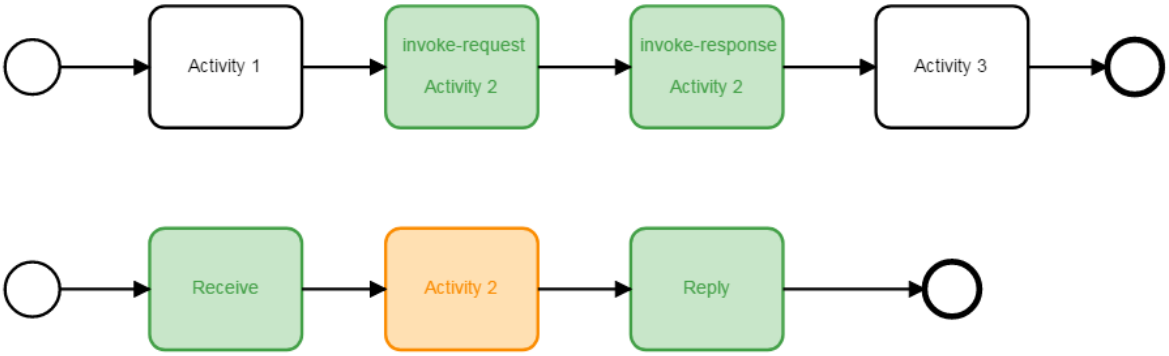

Figure 3.7. Synchronous communication

In the case of asynchronous communication, the main process invokes the other process as in synchronous case, but it does not wait until the other process finishes and continues its execution. This is shown in figure 3.8.

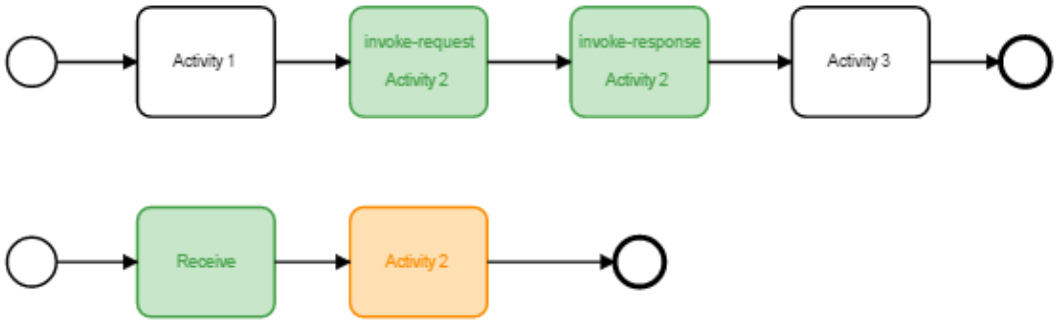

Figure 3.8. Asynchronous communication

3.3.1 Implementation with Camunda 7.8

In this subsection the Camunda business process management and workflow engine platform was chosen to implement the model decomposition process that was proposed in the last chapter. The chapter gives a good overview about how Camunda platform fits to achieve the goal, what are the limitations and possible workarounds. As the proposed approach was designed to use vanilla BPMN2.0 standard, one of the goals here is to use only Camunda and as less third party libraries as possible.

Camunda was chosen based on circumstances that the platform must be an open source, preferable lightweight and suitable for Java, there should be enough documentation and good community behind it. There are some good topics available in the Internet about possible options, for example in *Stackoverflow*. It is easy to see that Camunda is one of the best options.

3.3.1.1 Solution

The decomposition and distribution modules were packaged as a plugin for the Camunda engine so that so that it is easy to add to or remove from a Java based applications. Camunda engine can be installed as standalone web application or added as a Maven dependency for a Java application, and it works for Spring or Springboot setup too. The version of Camunda used for implementation is 7.8. It was the latest stable version at that time. The version of Springboot was 1.5.10 and the added Maven dependency version for Camunda business process management engine was 2.3.0.

It is possible to extend the Camunda engine with plugins [7]. A plugin must provide an implementation of the *ProcessEnginePlugin* interface.

The plugins can be added as event listener plugins. There are two types of event listeners: Task Event Listeners and Execution Event Listeners. Task Event listeners allow to react to Task Events such as when a task is *created, assigned* or *completed*. Execution Listeners allow to

react to events fired as execution progresses through the diagram: activities are *started*, *ended* and *transitions* are being taken [7].

There is less information about other plugin/listener types, but it is also possible to add a process engine plugin for BPMN parse listener and BPMN deploy listener. The most convenient plugin/listener for decomposition was a pre-deployer plugin. It is because intervening in a deploying process is the best way to change the model.

The source of the plugin code can be found in the repository: https://bitbucket.org/akiik/remote-executor/ .

3.3.1.2 System overview

As defined in the architecture, see diagram 3.1, there is a *master* and a *worker* node in this system. On both of these subsystems, the Camunda workflow engine is running. The master is the one with the extended plugin and is used by the user to deploy a business process. The worker is just an external engine where the master can offload extensive tasks that are marked as external during process design phase. This framework could be used in different situations, the low-power bottleneck is one such or for example, the case where the device can handle the tasks normally, but during sudden increases in load (maybe some other home automation processes started), not everything can longer be handled.

As said before, the modules were packaged as a plugin for the Camunda engine. The plugin is responsible for model decomposition and for offloading decomposed items to the worker. When this plugin is added to a Camunda engine, whenever user starts to deploy a new model to the engine, it checks every node of the process model and takes action if there is a node marked as external. To mark a node as external just a single property named “type” must be added to that node and it must be valued to “external”. If a deployed process model does not contain any node marked as external, the deployed model remains the same as original. The plugin does not require any third party libraries, but the Camunda connector plugin must be added to the engine since HTTP is used to communicate with a worker node.

3.3.1.3 The package overview

There are five classes in the plugin package:

- **BpmnRemoteSupportPreDeployerPlugin:** This is just a class for adding a custom pre-deployer plugin for an engine. It extends Camunda *AbstractProcesseningePlugin* class and overwrites the *preInit* method where the custom deployer is added.
- **BpmnRemoteSupportPreDeployer:** This is the deployer class itself which implements Camunda *Deployer* class. In the overwritten *deploy* method the decomposer instance is created, decomposing is called and in the end the original model is replaced with the extended one.

- **BpmnRemoteSupportDecomposer:** This class checks a process model and if it contains nodes marked as "external" it will decompose the process model so that the external node is replaced with *SendTask* and *RecieveTask*. The original task is encapsulated to a new process model definition which can be deployed to a worker. In addition to *SendTask* and *ReceiveTask*, the *Decomposer* adds XOR gateway and a *DelegationCheck* service task before that gateway. This process is described in the *3.3.1.5 The plugin in action* subsection later on.
- **DelegationCheck:** This class adds the flexibility to add some circumstances when to do the offloading and when not. This is up to the user what are these circumstances because this class can be easily overwritten, by default it uses CPU load to make the decision. The result is saved as a boolean process variable named as "delegate". In the figure 3.9 it will set the value "true" if CPU load is more than 50%.
- **SendTask:** *SendTask* is responsible for deploying the newly created process definition to a worker and for starting an instance from that process definition. It takes two HTTP requests to make this happen, one for deploying and one for starting an instance.

```
public class DelegationCheck implements JavaDelegate {
    @Override
    public void execute(DelegateExecution execution) throws Exception {

        OperatingSystemMXBean bean =
        (com.sun.management.OperatingSystemMXBean)
        ManagementFactory.getOperatingSystemMXBean();

        double load = bean.getProcessCpuLoad();

        execution.setVariable("delegate", load > 0.5);

    }
}
```

Figure 3.9. DelegationCheck class implementation

3.3.1.4 Adding a plugin to an engine

The official Camunda documentation says that the process engine configuration can be extended through process engine plugins. A process engine plugin is an extension to the process engine configuration. A plugin must provide an implementation of the *ProcessEnginePlugin* interface. There are several ways how process engine plugins can be configured depending mostly on the selected framework. In figure 3.10 an example of Spring boot plugin configuration is shown. This class must be added to the classpath. There is also a configuration for connector plugin since this is also needed.

```
@Configuration
public class MyConf {

    @ConditionalOnClass(ConnectProcessEnginePlugin.class)
    @Configuration
    static class ConnectConfiguration {

        @Bean
        @ConditionalOnMissingBean(name = "connectProcessEnginePlugin")
        public static ProcessEnginePlugin connectProcessEnginePlugin() {
            return new ConnectProcessEnginePlugin();
        }
    }

    @ConditionalOnClass(BpmnRemoteSupportPreDeployerPlugin.class)
    @Configuration
    static class RemoteConfiguration {

        @Bean
        @ConditionalOnMissingBean(name = "remoteProcessEnginePlugin")
        public static ProcessEnginePlugin remoteProcessEnginePlugin() {
            return new BpmnRemoteSupportPreDeployerPlugin();
        }
    }

}
```

Figure 3.10. Spring boot plugin configuration for Camunda engine

3.3.1.5 The plugin in action

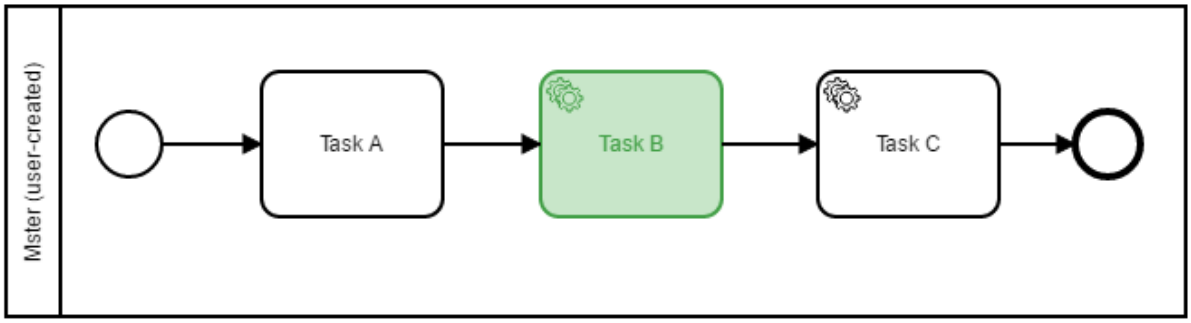

Figure 3.11. Process model before decomposition

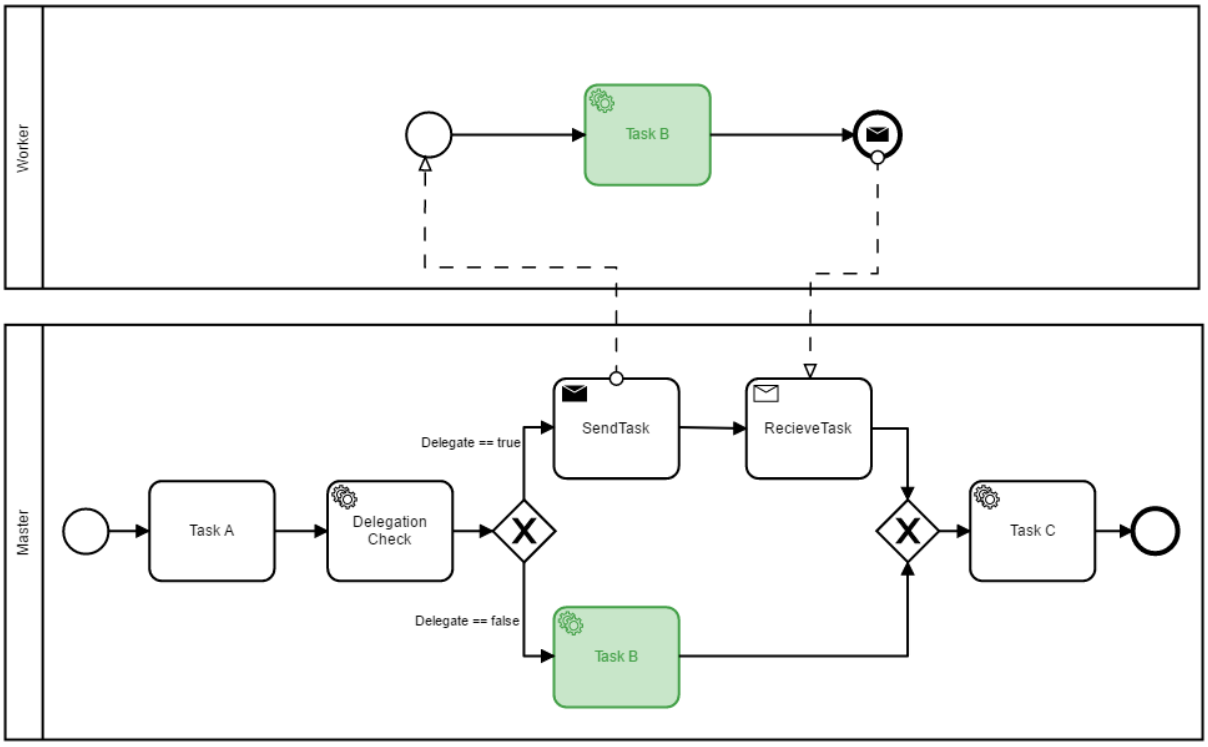

Figure 3.12. Process model after decomposition

Figure 3.11 and 3.12 show the transformations done by the implemented plugin. An example process model with three tasks was taken, shown in figure 3.11. The *Task B* is marked as external. After deployment to the Camunda engine where the plugin was used, the model was decomposed in the way as shown in figure 3.12 Master pool. When this process model instance is started, instead of executing *Task B* service task, first it will run the *DelegationCheck* task which returns a process variable named "delegate", the type of this variable is boolean and it is valued "true" if offloading is desired. After *DelegationCheck*, the execution reaches to a XOR gateway where the next path is chosen based on the value of "delegate" process variable. If it is *false*, it will execute the *Task B* and continue with *Task C*. If it is *true*, the execution runs *SendTask* node where the encapsulated *Task B* is deployed to a worker process engine and where it is immediately started. After *Task B* is started on a worker, the execution in *Master* pool will start the *RecieveTask* node. This node will wait a message with a result from worker engine. Once *Task B* finishes execution, the message is sent from the worker to the master engine. After that the process execution on the master engine can continue with *Task C*.

3.3.1.6 Additional requirements

It is important to know that the implemented plugin expects that a process which contains external tasks have process variables named *masterUrl* and *workerUrl* set. The *workerUrl* is used for referring an external entity where to the offloading should be done and the *masterUrl*

is used for referring a master engine where the master process is running. It is also required that the task which is marked as external is asynchronous, since when starting a process instance through Camunda REST API, it gives a response for that request when the process is done. That is not acceptable since the main process in the master engine must continue execution and reach to *ReceiveTask* where it waits a result from worker.

All these process variables that are used by plugin should be considered as reserved and should only be used in the process for the purpose defined by the plugin. There is also a *result* variable that is used for returning a result from the worker.

It is possible to start a process instance with process variables included through Camunda Process Engine API or REST API. In the figure 3.13, the REST API example is demonstrated.

```
HTTP POST: {{url}}/process-definition/key/Script_process/start

BODY:

{
	"variables": {
	"workerUrl": {"value": "http://192.168.43.1:8080/rest", "type": "String"},
	"masterUrl": {"value": "http://192.168.43.2:8080/rest", "type": "String"}
	},
	"businessKey": ""
}
```

Figure 3.13. An example Camunda REST API request to start a process instance with variables.

4 Evaluation

In this section, to evaluate the performance and scalability of the implemented system, two case studies were conducted. First, a real-life inspired smart door lock scenario, and second case study focuses at the scalability using compute intensive tasks.

4.1 Experiment Scenario

The experiment scenario is a simplified version of a smart door lock system described in [18]. There are only two activities in this process, the first one is for face detection and recognition while the second one is just for displaying the result. The process model is shown in figure 4.1 and the full XML of this model is included in Appendix I.

In Camunda engine, a Script Task is an automated activity. When a process execution arrives at the Script Task, the corresponding script inside the process definition XML is executed. This adds flexibility to run some code without implementing a class as required by service task. That is why this type of task was used and Groovy was chosen as a scripting language.

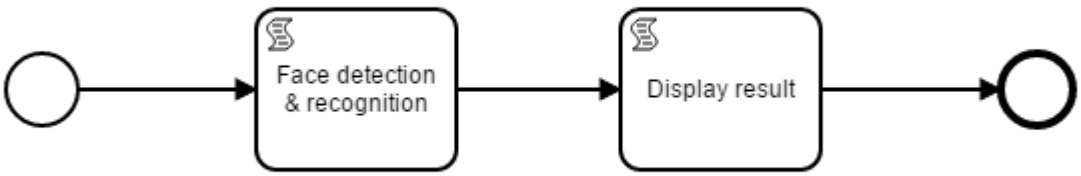

Figure 4.1 Experiment scenario ”Smart Door Lock” process model.

For face detection and recognition a tool created by Adam Geitgey was used [23]. It provides a simple face recognition command line tool that lets you do face recognition on a folder of images from the command line. There are two input folders it needs. First one is a folder with one picture of each person who are known. There should be one image file for each person with the files named according to who is in the picture, for example *Andres.jpg*. The second folder is for the files that needs to be identified.

In this implementation, the command invoked using Groovy is:

face_recognition ./known_people/ ./unknown_people/

4.2 Experiment Environment Setup

The goal was to imitate a typical smart home system, where a home controller system runs on a tiny and affordable device such as Raspberry Pi. For offloading, there must be something more powerful, usually it would be an external server. For the experiment, a Raspberry Pi 3b was chosen to run the home controller system, considered as a *master*, and *worker* runs on HP EliteBook 840 Intel Core i5-5200 2.2GHz 8GB RAM WIN 8.1 Enterprise. These devices are connected to each other with WIFI connection through LG Nexus 5X hotspot.

4.3 Experiment Pictures

For simplicity, only one image was used to connect a face with name and the other one was for identifying a face. A resolution for these pictures was 4000x3000 pixels.

Picture 4.1 Pictures used for learning a face name (left) and for detecting a face (right).

4.4 Experiment with and without offloading

The results are shown in figure 4.3, it took 49.44 seconds to run the process model locally while with offloading it took almost three times less time, 18.09 seconds. This shows that the offloading functionality that the implemented plugin adds is worth to consider. It is important to keep in mind that an ordinary laptop was used for offloading, therefore having a better resource would increase the benefit gained from the offloading.

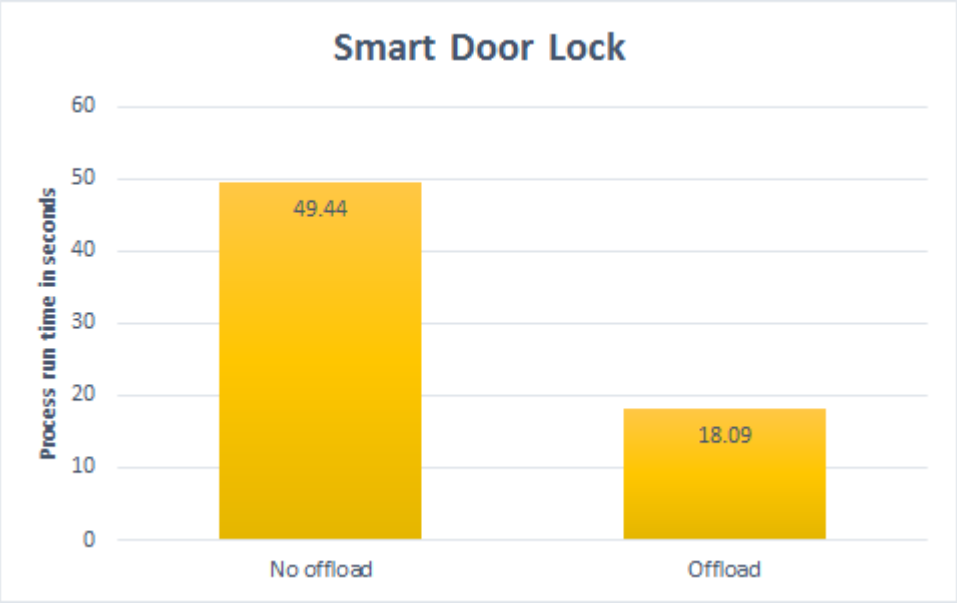

Fig. 4.3 Smart Door Lock process with and without offloading.

4.5 Scalability testing

In this 2nd case study, the goal is to see the scalability in case of differing task sizes (in terms of computation) and the influence of concurrency (in terms of multiple process instances). Overhead is in general referred to resources that are used but do not contribute directly to the intended result. Talking about the implemented plugin, HTTP communication should be considered as an overhead. It is highly dependent on a connection quality between master and worker system, a faster connection gives less overhead. Keeping this in mind, if a task is small enough it might not make sense to offload it. To get a better idea about the overhead level caused by offloading, some tests were performed. Since a face detection is a quite heavy task, it was not suitable for testing the overhead. A good option would be to use something simple and scalable, like a sorting algorithm. The chosen algorithm is shown in figure 4.4, it is made by Phil Cohen [24]. The task run time is correlated with the array size used for sorting. The overhead was tested with array sizes 100, 1000 and 10000. The numbers for the array were generated randomly beforehand and were hardcoded to task, that makes sure that the task complexity remains the same for all the executions, considering that sorting time might depend not only the size of an array but also on the order of the items in that array. The other variable that was taken into account, it is the number of instances. In business process management systems, several instances of a process model are running simultaneously and it would be interesting to know how the overhead will change while the number of instances is changing.

The hardware configuration is the same as previously in the "Smart Door Lock" experiment, but the process is different, there is just a single Script Task into which the sorting algorithm is packed.

To start several process model instances simultaneously via Camunda REST API, Postman tool version 6.0.10 Collection Runner was used.

```
int[] data = [...]


for (int i = 0; i < data.length; i++) {


            // Find the minimum value.
            int minIndex = i;
            for (int j = i+1; j < data.length; j++)
                if (data[j] < data[minIndex])
                    minIndex = j;
```

```
            // Move it to the front.
            Util.swap(data, i, minIndex);
}

class Util {

   public static void swap(array, int indexOne, int indexTwo) {
        Object temp = array[indexOne];
        array[indexOne] = array[indexTwo];
        array[indexTwo] = temp;
   }

}
```

Figure 4.4 Sorting algorithm used for testing the overhead

The results are shown in table 4.1 and 4.2. It is easy to see that if the size of an array is 100 and it takes 0.16 seconds to execute that on a master, it is not worth it to offload this to a worker since with offloading it takes more time, 0.3 seconds on average. And even when running more instances in parallel, the situation is the same. The overhead is directly related with the number of instances, if number of instances is doubled then the overhead is doubled, for example, with 10 instances the overhead is ~0.3 seconds while with 20 instances it is ~0.6 seconds. This is visualised in diagram 4.1.

Table 4.1 Run time with no offloading

Nr of instances/ items on list	100	1000	10000
1	0.161	0.899	7.560
5	1.068	2.025	16.013
10	1.571	3.433	30.755
15	2.130	5.275	39.558
20	3.492	6.425	55.253

Table 4.2 Run time with offloading

Nr of instances/ items on list	100.00	1000	10000
1	0.321	0.284	1.247
5	1.189	1.137	4.282
10	1.825	2.742	8.149
15	2.499	3.128	10.694
20	4.140	3.347	14.58

On the other hand, if there are 1000 items on the array to sort, it is already beneficial to use offloading, because with one instance the execution time on a master takes 0.9 seconds while with offloading it takes 0.28 seconds and the more instances runs the bigger the difference is. The correlation between run time and number of instances is the same as previously, more instances means bigger difference in run time.

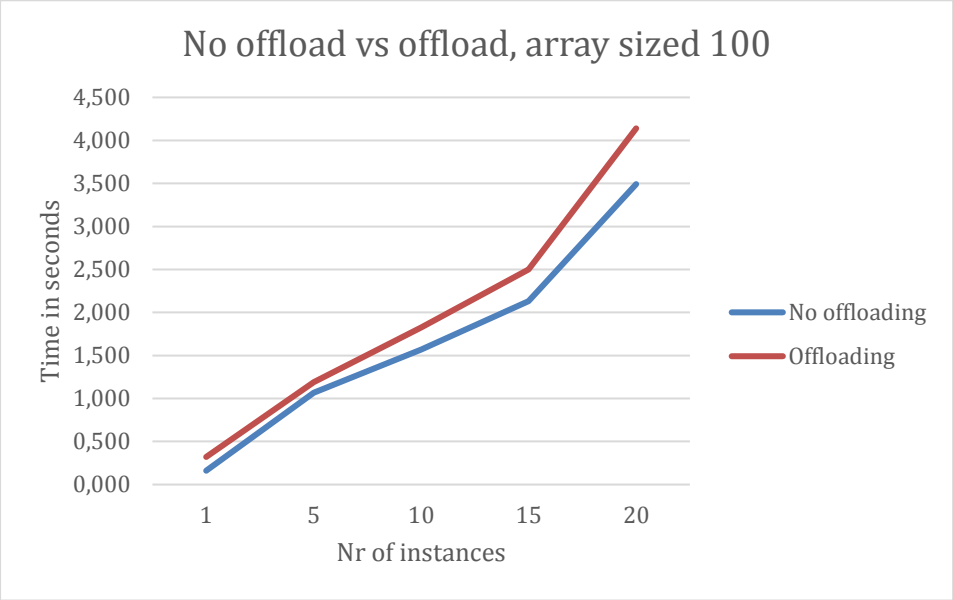

Diagram 4.1 Offload testing with array of size 100

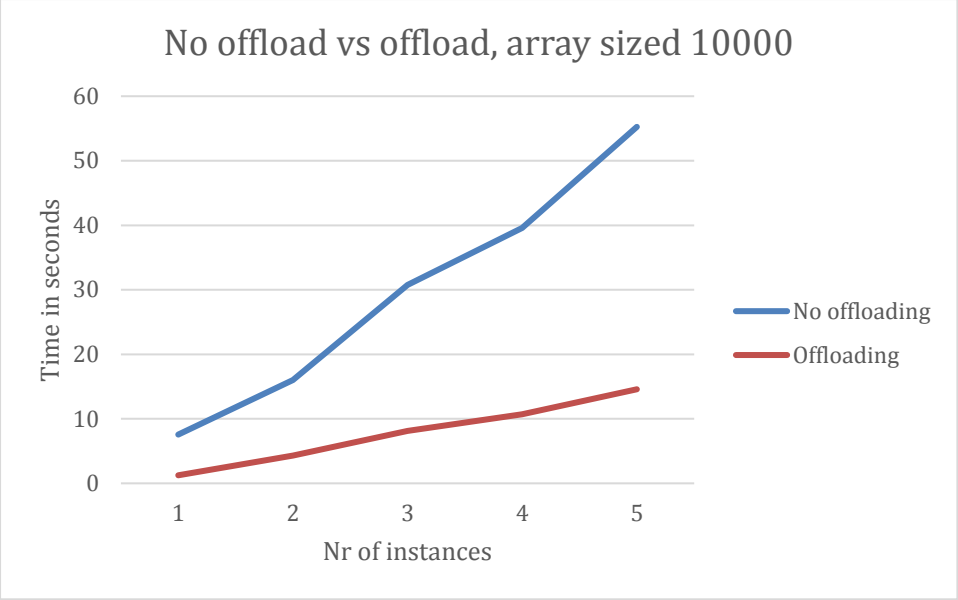

Diagram 4.2 Offload testing with array of size 10000

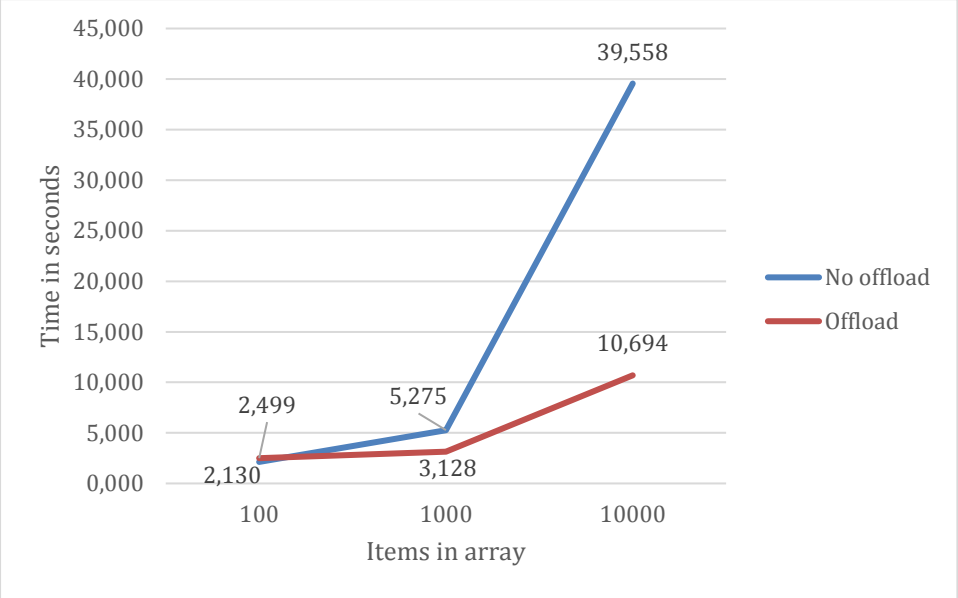

Diagram 4.3 Offload testing depending on the size of an array

As expected, if the complexity of a task rises, the more it makes sense to use offloading. With 10000 items in an array, the difference running one instance with and without offloading is ~6.25 seconds. It is shown in diagram 4.2.

A conclusive diagram 4.3 gives the best idea about the overhead level caused by this plugin and more importantly, about the big benefit that one can gain using this plugin.

5 Conclusion

In this thesis, a framework for adaptive execution of BP in IoT, capable of automated decomposition and offloading was presented. As a foundation for this framework, a literature review about IoT-aware business processes was done. The proposed framework enables offloading business process parts to an external entity by means of process decomposition. There are three main components in this system: *master*, *worker* and *IoT platform*. *Master* and *worker* are BPM workflow engines while *IoT platform* connects different home automation systems into one system. *Master* is the main workflow engine where a business process is running, *worker* is used for offloading. The implementation output is a plugin for a Camunda workflow engine. The plugin gives the opportunity to offload business process tasks to an external business process engine. The framework's integration with OpenHAB is also presented. The performance and scalability of the system was evaluated using two case studies. The results show that a significant benefit can be achieved when this framework is used, in the case of the smart door lock scenario for example, the run time was three times faster.

5.1 Future Work

The implemented plugin has some limitations such as the offloaded business process does not have an access to the original process resources, but sometimes there is a need that some or all process variables from the original model are accessible by decomposed process models too. Also, to support asynchronous business processes, some changes are needed. In addition to that, more value could be achieved if there is a possibility to add a specific offloading criteria to every task instead of one general criteria used by all tasks. For example, CPU intensive tasks could check the CPU load while network intensive tasks could check the network bandwidth instead.

6 References

[1] Dumas M., La Rosa M., Mendling J., Reijers H. A.. Fundamentals of Business Process Management. Berlin: Springer. 2013.

[2] Meyer, S., Ruppen, A.,Hilty, L.. The things of the internet of things in BPMN. In: Advanced Information Systems Engineering Workshops. Berlin: Springer. 2015, pp. 285–297.

[3] Gubbi J., Buyya R., Marusic S., Palaniswami M.. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems 29, 7 (2013), pp. 1645 – 1660.

[4] Evans D.. The internet of things: How the next evolution of the internet is changing everything. CISCO white paper 1. 2011.

[5] Object Management Group (OMG), Business Process Model and Notation (BPMN)

Version 2.0, 2011. http://www.omg.org/spec/BPMN/2.0/ (accessed June 30, 2011).

[6] Tan L., Wang N. Future internet: The Internet of Things. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, China. 2010.

[7] Homepage of Camunda. http://camunda.com/ (accessed 11.05.2018).

[8] Homepage of OpenHAB. https://www.openhab.org (accessed 11.05.2018).

[9] Kitchenham B., Charters S. M. Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report EBSE 2007-01. 2007.

[10] Sperner K., Meyer S., Magerkurth C.. Introducing Entity-Based Concepts to Business Process Modeling. In Dijkman R., Hofstetter J., Koehler J. Business Process Model and Notation. Berlin: Springer. 2011, pp. 166-171.

[11] Dar K., Taherkordi A., Baraki H., Eliassen F., Geihs K. A resource oriented integration architecture for the Internet of Things: A business process perspective. Pervasive and Mobile Computing, 20. 2015, pp. 145-159.

[12] Meyer S., Ruppen A., Magerkurth C. Internet Of Things-Aware Process Modeling: Integrating IoT Devices as Business Process Resources. In Salinesi C. Norrie M. C., Pastor Ó, Advanced Information Systems Engineering. Berlin: Springer. 2013, pp. 84-98.

[13] Ferreira P., Martinho R., Domingos D.. IoT-aware business processes for logistics: limitations of current approaches. 2010.

[14] Haller S., Magerkurth C.. The Real-time Enterprise: IoT-enabled Business Processes. In: IETF IAB Workshop on Interconnecting Smart Objects with the Internet. 2011.

[15] Meyer S., Sperner K., Magerkurth C., Pasquier J.. Towards modeling real-world aware business processes. In: Proceedings of the Second International Workshop on Web of Things. New York, NY, USA: ACM. 2011.

[16] Walewski, J. W. et al., Project Deliverable D1.2 - Initial Architectural Reference Model for IoT, 2011. http://www.iot-a.eu/arm/d1.2 (accessed November 4, 2016)

[17] Avilés-Lpez E., Garca-Macas J., TinySOA: A service-oriented architecture for wireless sensor networks, Serv. Oriented Comput. Appl. 3 (2). 2009, pp. 99–108.

[18] Igarashi Y., Hiltunen M., Joshi K., Schlichting R.. An Extensible Home Automation Architecture based on Cloud Offloading. In 2015 18th International Conference on Network-Based Information Systems (NBiS), Chengdu, China. 2015.

[19] Wieland M., Schwarz H., Breitenbücher U., Leymann F.. Towards Situation-Aware Adaptive Workflows. In: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), St. Louis, MO, USA. 2015.

[20] Motlagh N. H., Miloud B., Taleb T. UAV-Based IoT Platform: A Crowd Surveillance Use Case. In: IEEE Communications Magazine. 2017.

[21] Blinky LED with openHAB on Raspberry Pi, https://mcuoneclipse.com/2015/12/24/blinky-led-with-OpenHAB-on-raspberry-pi (accessed May 11, 2018).

[22] Duipmans E., Pires L., Santos L.. A Transformation-Based Approach to Business Process Management in the Cloud. 2013, pp. 191-219.

[23] Face Recognition tool by Geitgey A., https://github.com/ageitgey/face_recognition (accessed April 13, 2018).

[24] Selection Sort Algorithm by Cohen P. https://github.com/phillco/Algorithms/blob/master/src/algorithms/SelectionSort.groovy (accessed May 13, 2018).

Appendix

I Experiment scenario process model XML

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:camunda="http://camunda.org/schema/1.0/bpmn"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="Definitions_1"
targetNamespace="http://bpmn.io/schema/bpmn" exporter="Camunda Modeler"
exporterVersion="1.10.0">
  <bpmn:process id="Script_process" name="script" isExecutable="true">
    <bpmn:startEvent id="StartEvent_1">
      <bpmn:outgoing>SequenceFlow_0315cwu</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:sequenceFlow id="SequenceFlow_0315cwu" sourceRef="StartEvent_1"
targetRef="Task_0x49yt2" />
    <bpmn:sequenceFlow id="SequenceFlow_1sfnjx5" sourceRef="Task_0x49yt2"
targetRef="EndEvent_081jlr4" />
    <bpmn:scriptTask id="Task_0x49yt2" scriptFormat="groovy"
camunda:resultVariable="result">
      <bpmn:extensionElements>
        <camunda:properties>
          <camunda:property name="type" value="external" />
        </camunda:properties>
      </bpmn:extensionElements>
      <bpmn:incoming>SequenceFlow_0315cwu</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0dkgvva</bpmn:outgoing>
      <bpmn:script>
 'face_recognition /home/pi/known_people/ /home/pi/unknown_people'.execute().text
      </bpmn:script>
    </bpmn:scriptTask>
<bpmn:scriptTask id="Task_1hdpoxc" scriptFormat="groovy">
      <bpmn:incoming>SequenceFlow_0dkgvva</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1follyn</bpmn:outgoing>
      <bpmn:script>
       <![CDATA[import java.util.logging.Logger
       Logger logger = Logger.getLogger("result")
       logger.info('result =  '  + result)]]>
      </bpmn:script>
    </bpmn:scriptTask>
    <bpmn:endEvent id="EndEvent_081jlr4">
      <bpmn:incoming>SequenceFlow_1follyn</bpmn:incoming>
    </bpmn:endEvent>
  </bpmn:process>
</bpmn:definitions>
```

II License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kiik, Andres**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

    1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

    1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Business Process Decomposition & Distribution for Adaptive Internet of Things,

supervised by Mass, Jakob

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **21.05.2018**