MIKA SAARI

Software Hardware Combination for IoT Sensor Data Gathering and Prototyping

Architecture model, framework, and process model

Tampere University Dissertations 962

Tampere University Dissertations 962

MIKA SAARI

Software Hardware Combination for IoT Sensor Data Gathering and Prototyping

Architecture model, framework, and process model

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the auditorium 125
of the University Consortium of Pori, Pohjoisranta 11 A, Pori,
on 19 February 2024, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor and Custos	Associate Professor David Hästbacka Tampere University Finland	
Supervisor	Professor Kari Systä Tampere University Finland	
Pre-examiners	Professor Olov Schelén Luleå University of Technology Sweden	Dr. Niko Mäkitalo University of Helsinki Finland
Opponent	Professor Jari Porras LUT University Finland	

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Cover design: Roihu Inc.

ISBN 978-952-03-3306-5 (print)
ISBN 978-952-03-3307-2 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-3307-2

Carbon dioxide emissions from printing Tampere University dissertations have been compensated.

PunaMusta Oy – Yliopistopaino
Joensuu 2024

"The secret of getting ahead is getting started. The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and then starting on the first one."

Mark Twain

PREFACE

This work has been a long journey. The story began at Tampere University of Technology on June 1, 2002, when I was appointed as a research assistant for a programming course to assist Markku Nevanranta (thanks to him for his trust) in unraveling the mysteries of coding.

Over the years, debates with colleagues gradually increased the need to write my own doctoral thesis. I remember how my friend Sanna asked Professor Hannu Jaakkola at the post-doctoral celebration of Dr. Jari Soini in 2008: 'How long does it usually take to write a doctoral thesis?' – to which Professor Jaakkola replied: 'It will be completed when its time comes.' **Well, now its time has come**. I also want to express my gratitude to all three of you for your support.

The thesis work was conducted at two universities and in numerous faculties, which I cannot recall accurately due to numerous organizational changes. Throughout the entire doctoral project, I was employed as a university instructor by Tampere University of Technology, and by its successor, Tampere University, engaging in various programming teaching activities and interesting research projects. Financial support for this thesis was provided by the High Technology Foundation of Satakunta.

I had the privilege of being guided by Associate Professor David Hästbacka and Professor Kari Systä. Thanks to David for keeping the doctoral thesis process going and thanks to Kari for countless small clarifying questions. I remember best Kari's question at the beginning of the process: 'What is the benefit of this?' Well, here is the answer now. Additionally, I am very grateful to Professor Olov Schelén and Doctor Niko Mäkitalo for their excellent service as pre-examiners of this thesis.

I want to express my gratitude to my colleagues at Tampere University. Especially, I want to thank the large group of co-authors: Pekka, Petri, Haruka, Ahmad, Jere, Markku, Jaak, Sami, Timo, Janne, and Mikko. I also want to thank my friends (No - I still don't know anything about medicine) and all others who have helped

me on this long journey.

The importance of family cannot be overstated. Thanks to my parents for their support (sometimes even financial). And thanks to my children Eerika, Jessika, and Janette – is Dad going to become a doctor?

Finally, heartfelt thanks to my love, Niina. Your entrance at a time when the end seemed so distant brought new life and motivation. I am deeply grateful for your invaluable support in these final, defining moments.

Pori, February 2024
Mika Saari

ABSTRACT

Nowadays large scale data gathering is more common than previously, enabled by faster communication channels. In an Internet of Things (IoT) ecosystem, different kinds of sensor systems collect a huge amount of data from different environments. The first steps when starting to build a data gathering system are complex and challenging. This thesis provide guidelines for making a start. More specifically, this thesis describes data gathering with sensor devices and building a system.

In this thesis, the issue is approached using the design science method. At first, the problem was identified and divided into sub-questions. After determining the objective, the design of the prototype systems and development for data collection began. The developed systems were evaluated, documented, and the findings were published. During this thesis research work, more than ten prototype systems (most of which are discussed in the thesis) were built for gathering data from different environments.

The results and contributions of this thesis are divided into three IoT data gathering prototype development sections: sensor node architecture, a framework for IoT prototype development, and a process model for prototype development.

The sensor node architecture section introduces the abstract models developed for data gathering: multi node and single node architecture models. The main components are described: the sensor node, master node, communication, and the Internet as a communication channel for user applications. Furthermore, the purpose of the models and possible applications as a data collection tool are presented.

The framework section introduces the Software / Hardware (SW/HW) framework for IoT data collection. The framework categorizes prototype systems into three different types of construction, depending on the use case. Type 1 suits large amounts of data from a few sensor nodes. Type 2 collects simple data from several points with separate sensor nodes. In Type 3, smartphones are used as the data gathering sensor devices. In addition, several prototype applications with suitable

software and hardware components are presented.

The prototype development process section introduces the Descriptive Model for the Prototyping Process (DMPP). The process model brings together IoT prototype development practices that have been applied in research projects between university and enterprises.

These contributions were verified and validated by developing several data gathering prototypes. The architecture model and the framework were used in several prototype systems. The approach entailed setting a target, designing a measuring system, finding suitable tools, building the system, and evaluating the results. This prototyping took shape in similar processes, and was modeled into the descriptive model for the prototyping process itself.

The main results of the research and the thesis can be used as a guideline to make it easier to develop data gathering applications for an Internet of Things ecosystem.

CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

API	Application Programming Interface
Bluetooth	Short-range wireless technology standard used for exchanging data e.g., mobile devices
C/C++	C and C++ programming languages
CBSE	Component-based software engineering
CoAP	The Constrained Application Protocol (CoAP)
DS	Design Science
EEPROM	Electronically Eraseable Programmable Read-Only Memory
GPS	Global Positioning System
IDE	Integrated development environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIRA	Industrial Internet Reference Architecture
IoT	Internet of Things
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	Telecommunication standardization sector of ITU
Java	Java programming language
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
OS	Operating system

PHY	Physical Layer
Python	Python programming language
RAM	Random Access Memory
REST	REpresentational State Transfer
ROI	Return on investment
SBC	Single-Board Computer
SBM	Single-board microcontroller
SLR	Systematic Literature Review
SN	Sensor Network
SRAM	Static random-access memory
SW/HW Framework	Software/Harware Framework
UI	User Interface
USB	Universal Serial Bus
WSN	Wireless Sensor Network
XBee	Brand name for wireless connectivity modules, based on the IEEE 802.15.4-2003 standard designed for point-to-point and star communications
XML	Extensible Markup Language
Zigbee	Zigbee is a low-power, low data rate, and close proximity (i.e., personal area) wireless ad hoc network

ORIGINAL PUBLICATIONS

Publication I Saari, M., Sillberg, P., Rantanen, P., Soini, J. and Fukai, H. (2015a). Data collector service - practical approach with embedded linux. *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1037–1041. DOI: 10.1109/MIPRO.2015.7160428.

Publication II Saari, M., Baharudin, A. M. bin, Sillberg, P., Rantanen, P. and Soini, J. (2016a). Embedded Linux controlled sensor network. *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 1185–1189. DOI: 10.1109/MIPRO.2016.7522319.

Publication III Sillberg, P., Saari, M., Grönman, J., Rantanen, P. and Kuusisto, M. (2020a). Interpretation, Modeling and Visualization of Crowdsourced Road Condition Data. *Intelligent Systems: Theory, Research and Innovation in Applications*. Ed. by R. Jardim-Goncalves, V. Sgurev, V. Jotsov and J. Kacprzyk. (Extension of peer reviewed conference publication Sillberg, Gronman, Rantanen, Saari and Kuusisto, 2018). Springer International Publishing, 99–119. ISBN: 978-3-030-38704-4. DOI: 10.1007/978-3-030-38704-4_5. URL: https://doi.org/10.1007/978-3-030-38704-4_5.

Publication IV Grönman, J., Rantanen, P., Saari, M., Sillberg, P. and Jaakkola, H. (2018a). Lessons learned from developing prototypes for customer complaint validation. *Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*. Vol. 2217, 27–30. ISBN: 9788670314733.

Publication V Saari, M., Sillberg, P., Grönman, J., Kuusisto, M., Rantanen, P., Jaakkola, H. and Henno, J. (2019a). Reducing Energy Consumption with IoT Prototyping. *Acta Polytechnica Hungarica* 16.9, SI, 73–91. ISSN: 1785-8860. DOI: 10.12700/APH.16.9.2019.9.5.

Publication VI Saari, M., Rantanen, P., Hyrynsalmi, S. and Hästbacka, D. (2022). Framework and Development Process for IoT Data Gathering. *Advances in Intelligent Systems Research and Innovation*. Ed. by V. Sgurev, V. Jotsov and J. Kacprzyk. (Extension of peer reviewed conference publication Saari, Rantanen and Hyrynsalmi, 2020). Springer International Publishing, 41–60. ISBN: 978-3-030-78124-8. DOI: 10.1007/978-3-030-78124-8_3. URL: https://doi.org/10.1007/978-3-030-78124-8_3.

Publication VII Saari, M., Soini, J., Grönman, J., Rantanen, P., Mäkinen, T. and Sillberg, P. (2020a). Modeling the Software Prototyping Process in a Research Context. *Information Modelling and Knowledge Bases XXXII*. Ed. by M. Tropmann-Frick, B. Thalheim, H. Jaakkola, Y. Kiyoki and N. Yoshida. Vol. 333. IOS Press, 107–118. ISBN: 9781643681405. DOI: 10.3233/FAIA200823. URL: http://ebooks.iospress.nl/doi/10.3233/FAIA200823.

Publication VIII Harjamäki, J., Saari, M., Nurminen, M., Rantanen, P., Soini, J. and Hästbacka, D. (2023). Lessons Learned from Collaborative Prototype Development Between University and Enterprises. *Proceedings of the 33th International Conference on Information Modelling and Knowledge Bases*. Ed. by T. Welzer Družovec, M. Hölbl, L. Nemec Zlatolas and S. Kuhar. University of Maribor, 273–300. DOI: https://doi.org/10.18690/um.feri.5.2023.13. URL: https://press.um.si/index.php/ump/catalog/view/785/1118/3128-2.

Author's contribution

The contribution of the author of this thesis to the included publications is listed here for the reader's convenience. The contribution of the publications in relation to

the thesis is divided into three topics. Chapter 3 focuses on the topics of Publications I – III. Chapter 4 focuses on the topics of Publications IV – VI. And finally, Chapter 5 focuses on the process modeling topic of Publications VI, VII and VIII.

Publication I — The study consists of the lessons learned from the first experiment of the data gathering prototype. The primary task of the author was to design and implement the prototype system. One of the author's contribution was the background study with a literature survey. Co-authors gave valuable feedback and help with first prototype implementation. The author was the principal author of the publication and presented the work at the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) conference in May 2015.

Publication II — The study presents an extension of the data gathering prototype described in Publication I. The primary task of the author was to design a model for a data collection node in the sensor network. In addition, the author, along with co-authors, participated in the implementation of the system based on the model. The study introduces the sensor network multi node architecture model. The author was the principal author of the publication. The author of this thesis presented the work at the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) conference in May 2016.

Publication III — The article introduces a combination of models for data gathering and analysis of the gathered data, enabling effective data processing of large data sets. The content and structure of the article were planned together with the authors. The author's contribution was related to Section 3.1 "Data Gathering" including the data collection model and its modification into a single node type solution. Pekka Sillberg's contribution was related to Section 3.2 "Data Processing: Manageable Data Sources". The authors together discussed and constructed the entirety. The article is an extension of peer reviewed conference publication Sillberg

et al., 2018 which was presented at the International Conference on Intelligent Systems (IS) in September 2018 by author.

Publication IV — The study introduces two prototypes installed in vehicles and a cloud service for the autonomous collecting of data. The prototypes utilized a camera, location data, and timestamps for use cases. The first prototype was implemented for the Android mobile platform and the second one for the Raspberry Pi single-board computer. The study presents the differences and challenges faced in designing and implementing two prototypes for different platforms. The author contributed with the earlier developed data gathering node models and participated in the design and implementation of the prototype systems. The prototype system was developed and tested with the co-authors.

Publication V — The study provides knowledge about the usage of open hardware, open software, and open architectures with the development of prototype systems for gathering data on the research subjects of energy saving in the area of real estate and housing. The author's contribution was to introduce the ideas about an Internet of Things (IoT) prototype development framework. The other co-authors give valuable feedback to improve and revise the study. The author was the principal author of the publication and the article was an extension of a study by Saari et al., 2019b.

Publication VI — The article introduces a special software/hardware framework for data gathering systems to be used in IoT related systems. Also, the development process for IoT data gathering is introduced. The software/hardware framework was created by the author alone. Co-authors gave valuable feedback to improve and revise the article. The article is an extension of peer reviewed conference publication Saari, Rantanen and Hyrynsalmi, 2020 which was presented at the IEEE International Conference on Intelligent Systems: Methodology, Models, Applications in Emerging Technologies in August 2020 by author.

Publication VII The article presents an implementation of university – enterprise collaboration in prototype development described by means of process modeling notation. The focus is on modeling the software prototyping process in a research context. The author was the principal author of the publication and co-authors give valuable feedback to improve and revise the article. The paper was presented by Jere Grönman at EJC 2020, the 30th International Conference on Information Modelling and Knowledge Bases in June 2020.

Publication VIII The article presents an implementation of university – enterprise collaboration in prototype development. It focuses on the present usability and evaluation of the DMPP model in the KIEMI project. The author contributed to evaluating the suitability of DMPP in this kind of research environment. Janne Harjamäki contributed with his knowledge the enterprise collaboration the related aspects of research. The other co-authors give valuable feedback to improve and revise the article. The author of this thesis presented the work at EJC 2023, the 33rd International Conference on Information Modelling and Knowledge Bases in June 2023.

1 INTRODUCTION

Nowadays data gathering is very common. Commercial sensor systems of different kinds collect a huge amount of data from different environments. This thesis presents guidelines for building sensor devices for various environments. These guidelines were tested with several data gathering prototype systems in different projects carried out at Tampere University.

This thesis follows the research timeline. First, a wireless sensor network for data gathering is examined. This is done with prototype systems where the sensor node plays the main role. Also, the architecture model for a data gathering sensor node is presented. Second, the development of the architecture model into a sensor node framework is continued. The architecture model is an abstract tool for selecting different components and the framework is a more detailed approach, which gives examples of possible existing product components. The framework consists of software and hardware components, and is tested with the data gathering prototype systems that are developed. Third, the focus changes back to modeling – modeling of the development process. This is done by selecting four prototype systems, identifying their main factors, and modeling the process. Thus, the last part of the thesis focuses on the model of the prototype system development process.

1.1 Motivation

The Internet of Things (IoT) refers to the expansion of Internet services, connecting everyday physical objects to a network. This connection between a network and physical objects makes it possible for the things (devices) to interact with each other and cooperate with their neighbors to achieve common goals (Atzori, Iera and Morabito, 2010). According to predictions made by GSMA Intelligence (GSMA Intelligence, 2020), there will be about 24 billion IoT devices by 2025.

The first mention of the term 'IoT' is said to have come from Kevin Ashton in

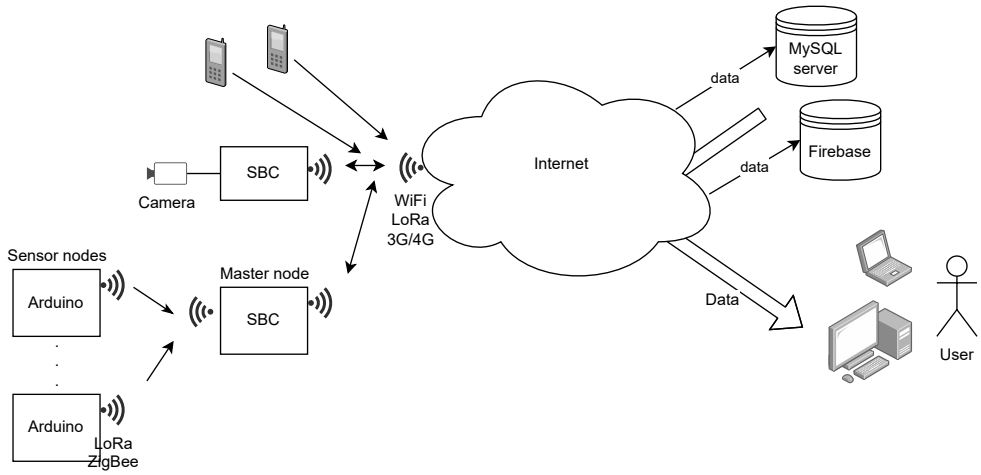

Figure 1.1 The big picture. A few technologies or components commonly used in the thesis have been added to the figure.

1999 (Ashton, 2009). Several research papers have been written about IoT and what it includes. Barr (2020) defines IoT as "including mobile devices to vehicles, home appliances and other products that connect, communicate, and exchange data over the Internet." A survey of the areas of “Internet of Things” was made by Atzori, Iera and Morabito (2010). The basic features of sensor networks were compiled in a survey conducted in Akyildiz, Sankarasubramaniam and Cayirci (2002).

Despite the above-mentioned studies, IoT could be understood to mean several types of systems. In this thesis, the term IoT refers to the system illustrated in Figure 1.1 where it consists of sensors, the data gathering modules, data connection, the Internet capable technologies to transfer data, and data storage, the cloud system to store data. The users are provided with the stored data.

This research work originated from the idea of gathering data and using the gathered data to deliver information to interested parties. There are several threats in a world where information could save human lives - threats like tsunamis, volcanic eruptions, and earthquakes. In Finland, threats of this kind are not part of everyday life, but there are areas, like Japan, where they are. This research started in collaboration with Keio University. They had an interest in using the ideas we developed in the area of delivering of emergency messages using mobile phones (Sillberg et al., 2009).

Thus, the starting point of the research began with the idea of gathering data for

an emergency alert system. The idea was to collect data from as many points as possible, analyze the data, and try to recognize anomalies. Furthermore, anomalies would cause an alert and the alert messages would be sent to the people in the area where the anomalies occurred. (Publication I)

At the beginning of the research process, other possibilities for utilizing IoT information in the university environment were identified. IoT research in universities is quite wide, starting from IoT teaching with the devices by testing and programming (Saari et al., 2015b). The ability to collect, analyze, and use IoT data offers a large amount of research targets, which are highlighted in studies by Saari, Baharudin and Hyrynsalmi (2017) and a study focused more on communication technology by Saari et al. (2018).

The terms model and modeling are mentioned in this research several times. This shows the influence of the author's learning and research environment in the Pori unit of Tampere University. Jaakkola, Henno and Thalheim (2016) raised the question "Why Information System Modeling Is Difficult?" The thesis clarifies the modeling issues given below, starting with the research questions.

1.2 Research questions

This thesis claims that it is reasonable to combine wireless sensor network (WSN) node modeling, a framework for prototypes, and a process model of prototype development. The claim is based on the finding of this study that all these aspects are needed to build the data collection prototype system to achieve a successful outcome.

Based on these considerations, the following research question was defined:

How to construct a system architecture model of wireless sensor network nodes and process models for the prototyping process to efficiently develop data gathering for IoT applications?

This sentence combines the main components of the research focus. This main question can be divided into three sub-questions:

RQ1: What kind of sensor model architecture can be developed for data gathering in a wireless sensor network?

RQ2: How can IoT data gathering be generalized into a framework of required software and hardware components?

RQ3: What kind of process model can be developed from prototype development practices that have been applied in research projects between university and enterprises?

The first sub-question was based on the following hypothesis: It is possible to generate a multi-purpose sensor node model for the gathering, pre-processing, and storage of different kinds of data. This multi-purpose model enables faster system development and provides guidelines for the design of the internal structure of sensors. The focus of the hypothesis was to construct an architecture model for a WSN sensor node that combines software and hardware components.

Further, the hypothesis was tested with several prototypes in which the hardware and software used were described. The hardware components were off-the-shelf (or in other words "ready-to-use") devices such as a single-board computer (SBC), smartphone, single-board microcontroller (SBM), communication modules, sensor modules, power supplies and similar. The software components were open source or free to use software, and some devices also included dedicated software. These topics are handled and tested with prototype systems in more detail in Publications I-III. The objectives and answers to the research questions are clarified in Chapter 3 of this thesis.

The second sub-question was constructed from the following hypothesis: It is possible to generalize the software and hardware components used from IoT and data gathering prototypes into a framework that can be used in many ways for different purposes (e.g., for energy, heating, and electricity savings (Publication V). The framework for data gathering is a combination of software and hardware, which is the focus of Publication VI. The answers to these sub-questions are presented in Chapter 4.

The third sub-question was the conclusion of previous research and based on the following hypothesis: There are prototype development practices that can be modeled in a process model for use in collaboration between research institutes and enterprises. This topic was researched using earlier prototype development processes in university - enterprise collaboration and by identifying the main factors, which were role, activity, resource, and artifact. The development process is modeled in Publication VII; its implementation is described in Publication VI. Publication VIII

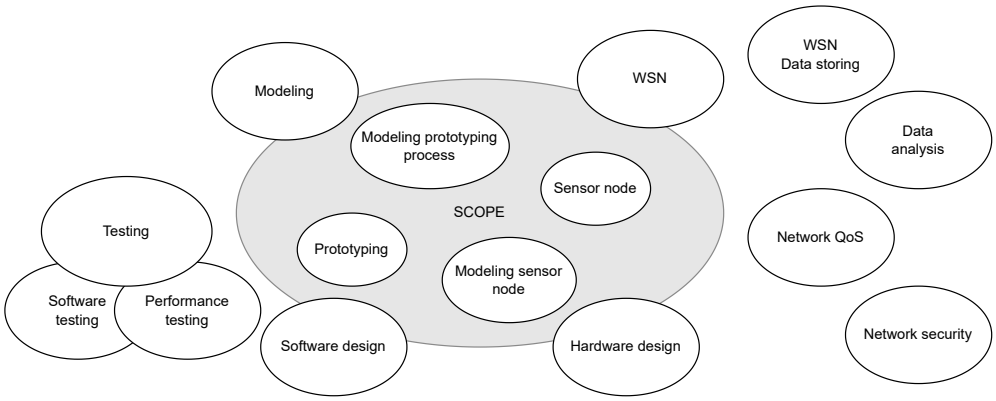

Figure 1.2 Thesis scope.

evaluates the usability of the developed model in a university research project. The process and the model are presented in Chapter 5.

1.3 Scope and contributions

The scope of the thesis concentrates on selected IoT-related topics (1.2). The scope of the thesis (blue ellipse) is centered on the sensor node and its modeling. Several prototypes were made and finally the development process itself was modeled.

In addition, four major themes were involved: modeling, WSN, software design and hardware design. These themes are only partially addressed in the thesis, depending on how each of them was needed to complement the prior knowledge. These topics are clarified in Chapter 2 with related studies. The topics outside of the blue ellipse are not covered in detail in this thesis. However, the out of scope topics were handled in the research papers that support the concept of the thesis.

Several IoT-related applications were created during the study. The research theorem and research questions were defined in relation to these data gathering prototype applications. The answers were found to the research questions; the main contributions and results of the thesis are as follows:

- The **Architecture model** refers to the SN or WSN data gathering sensor node architecture. The architecture consists of software and hardware components, and their interconnections on an abstract level. The two architecture models developed for the sensor node are presented in the first three Publications I-III

and Chapter 3.

- The **Software / Hardware (SW/HW) framework** generalizes prototype development into groups of required software and hardware components; more precisely, the framework defines the guidelines for constructing prototype systems to collect data for different purposes. Publications IV-VI and Chapter 4 focus on the SW/HW framework.

- The **Descriptive Model for the Prototyping Process (DMPP)** is the process model developed from the prototype development practices applied in research projects between the university and certain enterprises. Publication VII introduces the modeling of the process and presents the DMPP. Further, Publication VI combines the DMPP with the context of the SW/HW framework. The last publication, Publication VIII, evaluates the usability of the developed model in a university research project. The DMPP is presented in Chapter 5.

In this thesis, the DMPP combines the developed architecture model and the framework with the prototype development process model (Figure 1.3). In the first phase (1. Define Requirements) decisions must be made by the developer team: What do we want to measure? After this decision, a suitable architecture model for the data is selected. This provides the basic requirements for the prototype system (2. Requirement Notes). The selection of the architecture model can be used as a guide when selecting a suitable framework of components to construct the first prototype in the development phase (3. Develop Prototype). The fourth phase of the DMPP (4. Develop Artifacts) includes the working prototype and the results of its usage.

In the context of this thesis, the phase is completed when the prototype collects data and the data meet the requirements of phase two. The fifth phase (5. Prepare and Conduct Presentation) is the preparation to present the prototype system, collected data, and data analysis to the project partners. The sixth phase is the publication or release of the developed prototype system.

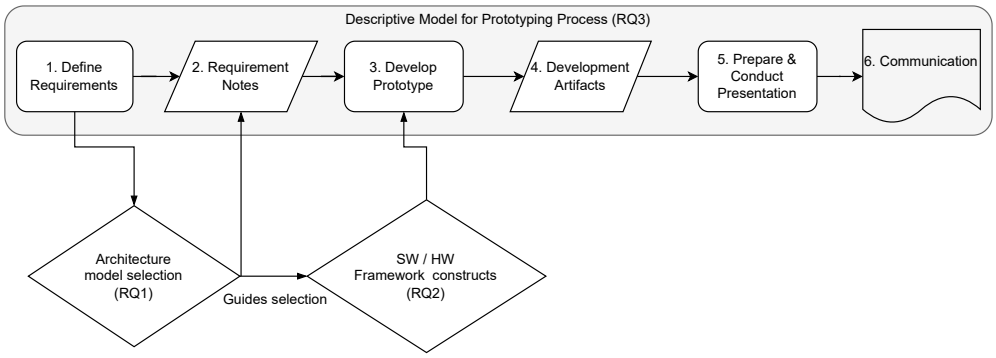

Figure 1.3 Thesis research questions (RQ) 1-3, contributions and their interconnections.

Table 1.1 Research questions

Research questions	Publications	Thesis chapter
RQ1: Architecture model of WSN data gathering sensor node	I, II, III	3
RQ2: SW/HW framework for IoT data gathering prototype	IV, V, VI	4
RQ3: Descriptive model of the prototyping process (DMPP)	VI, VII, VIII	5

1.4 Research methodology

The research method used in the thesis is Design Science (DS), which is a practical constructive research method in the information systems field. The analysis is mostly qualitative because of the difficulties involved in proper quantitative analysis. Despite this, quantitative analysis is used partially in some use cases, for example in Publication III, where the prototype system produced a large amount of numerical data.

DS and its methodologies are explored extensively in Nunamaker, Chen and Purdin (1991), Hevner et al. (2004), and Peffers et al. (2007). In DS, the attempt is to create artifacts to serve human purposes. Peffers et al. (2007) have stated: "The development of the artifact should be a search process that draws from existing theories and knowledge to come up with a solution to a defined problem". The DS process includes six steps: problem identification and motivation, the definition of the objectives for a solution, design and development, demonstration, evaluation, and

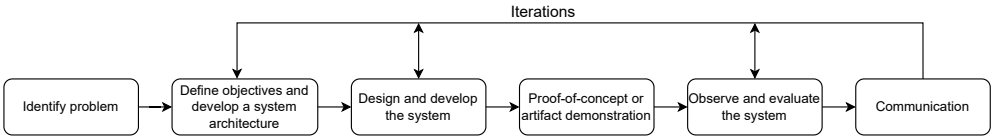

Figure 1.4 Process for system development research in this thesis. Adapted from Nunamaker, Chen and Purdin (1991) and Hevner et al. (2004)

communication (Peffers et al., 2007). These six phases were adapted as guidelines to be followed in this research and thesis work. In addition, the research method adopted makes it possible to use iteration rounds during the research, which are also illustrated in Figure 1.4.

The first phase "Identify problem" is the main research question: "How to construct a system architecture model of wireless sensor network nodes and process models for the prototyping process to efficiently develop data gathering for IoT applications?" The conceptual framework consists of modeling, WSN, and IoT. These top-level concepts delimit the field of study.

In the next phase "Define objectives and develop a system architecture", the main research question was used and the specific terms were taken with the top-level concepts: constructing a model for a WSN sensor node. This gave rise to the first sub-question: "**RQ1:** Developing a sensor model architecture for data gathering." The selected research method enables iterations within the development process, which is illustrated in Figure 1.4. The iteration rounds produced sub-questions "**RQ2:** SW/HW framework for IoT data gathering prototype" and "**RQ3:** Descriptive model of the prototyping process (DMPP)".

The purpose of the third phase "Design and develop the system" is to produce knowledge for the base schema. In the scope of this thesis, this knowledge is collected in the "related studies" sections of the publications. During the research, the knowledge was collected by using the systematic literature review (SLR) method introduced by Kitchenham and Charters (2007). The SLR method was used and the results are presented in Publication V and Saari, Baharudin and Hyrynsalmi (2017). The second method used, introduced by Petersen, Vakkalanka and Kuzniarz (2015), is a systematic mapping study to give insight into the research trends within the scope of the thesis. In addition, answers to the sub-questions of the thesis were sought in this phase.

The artifacts were built in the "Proof-of-concept or artifact demonstration" phase.

The term artifact in the scope of this thesis refers to any kind of tangible product produced during the development process. Further, Hevner et al. (2004) advised building artifacts within the "Design-Science Research Guidelines". These artifacts could be a model, method, or instance of the system. Publications I-VIII present the artifacts produced, which are summarized in Chapters 3 to 5. Chapter 3 presents architectural models for data gathering sensor node artifacts. Chapter 4 describes the guidelines for creating a sensor node framework artifact for building new data gathering devices. Finally, Chapter 5 presents the process model artifact for developing data gathering prototypes.

The next phase, "Observe and evaluate the system," consists of testing. The first evaluation was made during prototype development, and some of the prototype systems proceeded to customer evaluation. The other way to test the results was to publish a scientific article, in which case the results were evaluated by academic reviewers.

In the final "Communication" phase, the knowledge and data are produced and published. This thesis is now the last communication artifact, although ideas for future research are presented in the conclusion chapter.

The iterations offer the possibility to develop the artifacts further, i.e., the data gathering prototype systems, architecture model, and the software/hardware framework. As an example: Publication I was the first to go through all the phases in the process. Publication II was the second iteration and Publication III was the third. The overall communication of the artifacts is provided in Publications I-VIII.

The evaluation of the research results has been subjected to international review by other researchers by publishing the results obtained. Furthermore, the review comments provided valuable information for the following iterations.

1.5 Thesis structure

The thesis structure in relation to the publications is illustrated in Figure 1.5. Chapter 2 contains a section, 2.1, on IoT architecture standards. After that, there are three sections to present existing work related to the research questions presented in Section 1.3. In addition, relevant background information, i.e., methods, techniques, and technologies, is provided.

Chapter 3 focuses on the **RQ1** sub-question and how Publications I-III relate to

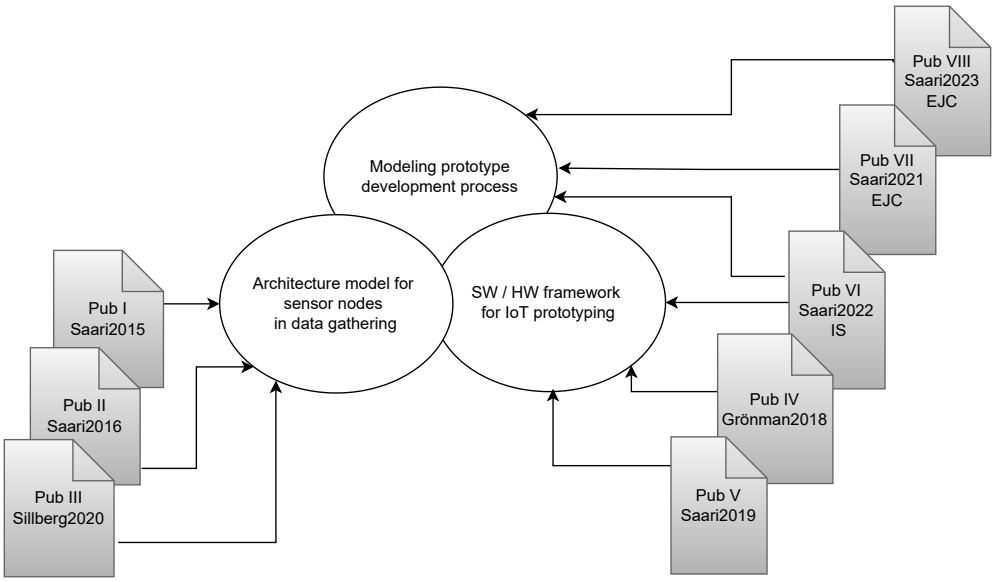

Figure 1.5 Thesis structure.

it. The main idea is to present the WSN node in two architecture models: **single node** and **multi node**. Chapter 4 continues the subject of generalizing prototype development into a software/hardware framework. **RQ2** sub-question is presented in the context of Publications IV-VI. Chapter 5 focuses on the modeling of the prototype development process, which is the subject of the last **RQ3** sub-question and discussed in Publications VI-VIII.

Chapter 6 concludes the thesis by summarizing the research. The research questions are revisited at the beginning of the chapter. The generalization of the research findings along with limitations are also dealt with. Finally, directions for future research are discussed.

2 BACKGROUND

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." Weiser (1991) said this in his article about "ubiquitous computing". IoT could be said to be environmentally aware computers that disappear into the background. This chapter starts with the introduction of IoT and related standards. The rest of the chapter is divided into three sections to present the existing work in relation to the research questions presented in Section 1.2.

2.1 Standards for IoT architecture

Guidelines and instructions are needed when creating a system architecture. Standard "ISO/IEC/IEEE 42010:2011, Systems and software engineering — Architecture description" addresses the creation, analysis, and sustainment of architectures of systems through the use of architecture descriptions. It also includes a core ontology for the description of architectures. The ISO/IEC/IEEE 42010:2011 standard is used as a reference when creating or presenting architecture related standards.

The architecture of the IoT system can be represented as layers. The IoT system can be divided into three layers: device, network, and application. The device collects the data. The application is the provider of the data. Between these, the network is the technology which moves the data from the device to the application. The abstract architecture sets requirements for the properties of each layer but does not specify the exact equipment. This abstract approach creates a foundation of component-based development when each component is a higher-level abstraction. In software engineering, component-based software engineering (CBSE) is an important software development approach (Sommerville, 2016). Furthermore, a component is "an independently deliverable set of reusable services" and this information can be utilized in component-based development (CBD) (Brown and Short, 1997).

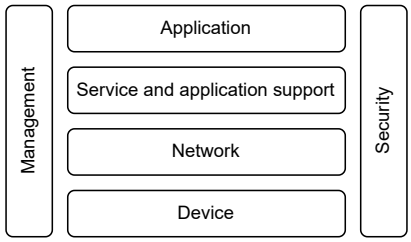

Figure 2.1 IoT reference model with four layers. Adapted from ITU-T Y.2060.

The three IoT standards (or related recommendations) included in this study are listed below. These are marked by their abbreviation, name, and date. The abbreviations are used later on this study to refer to a specific document.

- ITU-T Y.2060, Overview of the Internet of Things, 06/2012 (International Telecommunication Union, 2012)
- IEEE 2413-2019, IEEE Standard for an Architectural Framework for the Internet of Things, 05/2019 (*IEEE Standard for an Architectural Framework for the Internet of Things (IOT)* 2019).
- IIRA V1.9, The Industrial Internet of Things, Volume G1: Reference Architecture, 06/2019 (*The Industrial Internet of Things Volume G1: Reference Architecture, v1.9* 2019)

ITU-T Y.2060 is an overview of the IoT with clarification of concept and scope. IEEE 2413-2019 is a description of the architectural framework, which is aimed at stakeholders in IoT systems, for example: transport, healthcare, smart grid, etc. IIRA V 1.9 is a technical report, which describes the Industrial Internet Reference Architecture (IIRA) for Industrial Internet of Things (IIoT) systems. It provides guidance and assistance to Industrial Internet Consortium (IIC) members and the wider IoT community in the development, documentation, communication, and deployment of IIoT systems.

These two documents, IEEE 2413-2019 and IIRA V1.9, use the ISO / IEC / IEEE 42010: 2011 standard (*ISO/IEC/IEEE 42010:2011, Systems and software engineering — Architecture description* 2011) as a reference and furthermore utilize the architecture concepts it introduces. Also both IEEE 2413-2019 and IIRA V1.9 are based on the layer idea presented in ITU-T Y.2060 (Figure 2.1), but contain several extensions and refinements.

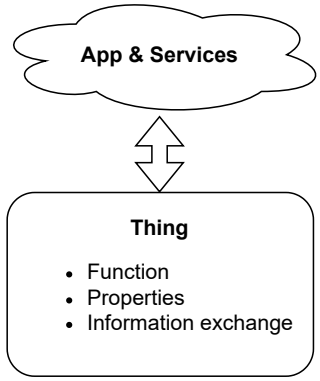

Figure 2.2 The "Thing" as a context of standard IEEE 2413-2019. Adapted from IEEE 2413-2019.

ITU-T Y.2060 - Overview of the Internet of Things was developed by the International Telecommunication Union (ITU) and is the oldest of the documents. The document is termed a recommendation and it provides an overview of the Internet of Things (IoT) with the main objective of highlighting this important area for future standardization. It includes IoT-related terms, concepts, features, and high-level requirements. In addition, the recommendation presents the IoT reference model (Figure 2.1).

The device layer can be thought of as a physical sensor device which observes the environment. The device has a network connection to communicate with the IoT service and an application support layer. The support layer contains IoT applications, such as data processing and data storage. The purpose of the application layer is to provide information to the users or clients of the IoT system.

The IoT reference model (Figure 2.1) also includes management and security capabilities in all layers. The management capabilities can be fault management, configuration management, accounting management, performance management, and security management. The security capabilities include authentication and authorization in the different layers.

IEEE 2413-2019 simplifies the layer presentation into two layers "Thing" and "Apps & Services" (Figure 2.2). "Thing" includes the device, information exchange, and its functions. The network layer exists, but is not specified more precisely. The terms "IoT system" (a system of entities including cyber-physical devices, information resources, and people) and "IoT environment" (IoT components that may be used to create the IoT systems) are clarified.

The architecture is presented as "the Abstract IoT Domain", which is the foundation of architecture frameworks and includes the common characteristics and behaviors of IoT systems. IEEE 2413-2019 presents representative domains: Smart Manufacturing, Smart Grid, Smart Buildings, Intelligent Transport, Smart Cities, and Healthcare, which are derived from the Abstract IoT Domain.

For this thesis, an important point, presented in IEEE 2413-2019, is a stakeholder concern "How can we make system concepts reusable, e.g., over product generations and/or across engineering teams?" This proves that there is still a need for the work described in the thesis.

IIRA V1.9 presents the industrial Internet architecture viewpoints in four levels: business, usage, functional, and implementation. The functional viewpoint is important for this research work: the components, their structure, and the interactions between them. The implementation viewpoint is also important when it deals with the technologies needed to implement functional components and their communication.

IIRA V1.9 is based on the same idea of layers (Figure 2.1), but there are several extensions. The implementation viewpoint presents three example architecture patterns:

- Three-tier architecture pattern
- Gateway-Mediated Edge Connectivity and Management architecture pattern
- Layered Databus pattern.

In terms of this thesis, these three patterns present "Edge-Tier" as a sensor or actuator tier, which collects data from edge nodes (Figure 2.3).

In summary, all three standards are worth knowing, as they all provide valuable background information on terms and concepts.

2.2 Current trends related to IoT

Central to this thesis is the concept of the Internet of Things (IoT), which can be envisioned as an advanced Wireless Sensor Network (WSN) that collects and processes data. Replacing traditional data-collecting sensors, edge computing processes data proximally to its source, minimizing the need for long-distance data transmission to

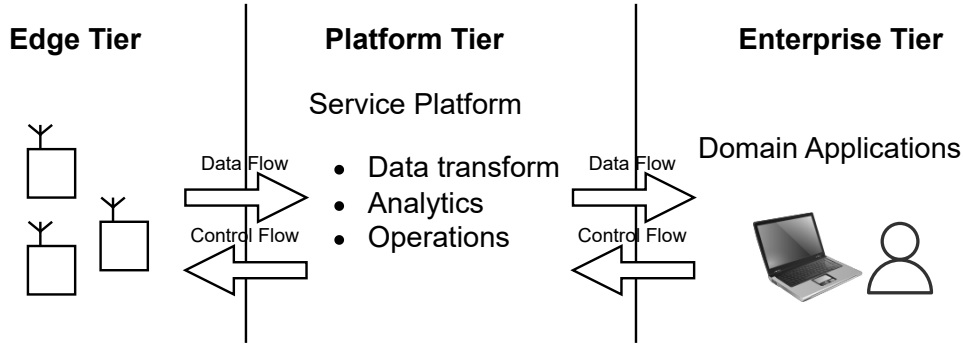

Figure 2.3 Three-Tier Industrial Internet of Things System Architecture. Adapted from IIRA V1.9.

centralized data centers or clouds. (Roman, Lopez and Mambo, 2018), (Hamdan, Ayyash and Almajali, 2020)

WSN communication technologies are pivotal in IoT ecosystems, facilitating data relay between sensors and platforms. Given the limited power resources of typical sensors, energy efficiency becomes a critical concern. To address this, innovative sensors are being developed to preprocess data on-site, significantly reducing energy consumption. (Gulati et al., 2022)

Furthermore, Artificial Intelligence (AI) plays an increasingly vital role in interpreting the voluminous datasets generated by IoT devices. AI applications can predict patterns, identify anomalies, and automate decision-making, enhancing the overall efficiency of IoT systems. (Nguyen-Duc et al., 2023)

Lastly, the interconnectivity inherent in IoT devices introduces numerous security challenges. It is imperative to implement stringent security protocols and encryption methods to safeguard sensitive data against unauthorized access and maintain the confidentiality and integrity of the IoT network (Atlam and Wills, 2020). More widely, the article by Porras et al. (2018) provides a comprehensive analysis of IoT security concerns and solutions, utilizing a manual systematic mapping study and automatic content analysis, identifying key challenges and research trends in IoT security.

2.3 Constructing a WSN sensor node architecture model

The architecture model of the data gathering sensor node 'Sensor networks' refers to distributed autonomous sensors used to monitor the physical environment, e.g., temperature or pressure. The main part of IoT systems is Sensor Networks (SN), especially Wireless Sensor Networks (WSN) (Perera et al., 2014), (Marković et al., 2016).

A sensor network is a group of sensor nodes for collecting physical environment data and send data to the data storage. Since sensor nodes have data processing ability, the uploaded data can be either raw or pre-processed (Akyildiz, Sankarasubramaniam and Cayirci, 2002). A WSN uses wireless network technologies for the communication of sensor nodes and sinks, which collect the data. A WSN includes sensor nodes, which consist of sensing, data processing, and communicating components.

During the research work of the thesis, three literature reviews were published: Saari, Baharudin and Hyrynsalmi (2017), Saari et al. (2018), and Saari, Nurminen and Rantanen (2022). The first two of these focus on resolving the state of research on IoT network sensor solutions. The third addresses the use of software components in IoT studies. These studies point to building prototypes as an important way to study the IoT environment.

A survey conducted in 2002 collected the basic design factors of a sensor network (Akyildiz, Sankarasubramaniam and Cayirci, 2002). Design factors can be used as a guide when developing prototype systems such as those presented in this thesis. The list below also mentions aspects and observed problems that were taken into account in the prototypes developed:

- Fault tolerance – a common fault is power outage and consequent problems such as data corruption.
- Scalability – the selected communication infrastructure might limit the amount of sensor nodes.
- Production costs – prototyping and testing the ideas should not be a large investment.
- Operating environment – open source with community support helps move the research forward.
- Sensor network topology – star topology was selected for testing.

- Hardware constraints – sensing unit, storage/processing unit, and transceiver unit.
- Transmission media – the prototypes use a feasible communication method.
- Power consumption – depends on the use case, battery use is avoided if possible.

A conceptual model is defined as "a representation of a system, made of the composition of concepts which are used to help people know, understand, or simulate a subject the model represents." In the computer science area, conceptual modeling concerns describing the semantics of software applications at a high level of abstraction (Embley and Thalheim, 2011). Furthermore, architecture refers to the fundamental structures when describing the system. A study by Al-Fuqaha et al. (2015) discusses the overall architecture of IoT and its elements by referencing several studies. In addition, the study presents four different scale models. The architecture model of IoT can be presented as a simplified three-layer construction of system structures, such as the sensing layer, network layer, and application layer. The sensing layer represents the physical sensors of IoT that aim to collect and process information. The network layer transfers data produced by the sensing layer to the application layer using various communication technologies. The application layer provides the information to the customer.(Al-Fuqaha et al., 2015)

There are several ways and reasons to model prototype systems in a WSN. An article by Galkin (2016) analyzes the different models of collecting information from WSNs. The data collection model is based on a suitable schedule for monitoring parameters. Jin et al. (2014) introduced the Physical Service model, where they separated a device model, resource model, and service model. Laukkarinen (2015) introduced a Wireless Sensor Network abstraction model with three levels: node abstraction, network abstraction, and infrastructure abstraction.

WSN sensor nodes can be divided into three categories: time-driven, event-driven, or query-driven (Barrenetxea et al., 2008). The time-driven system collects data periodically, for example, a temperature sensor measures the temperature every 10 seconds. In event-driven systems, data are collected if a particular event occurs (Publication IV). One example of this is a GPS-driven camera application. The query-driven system only sends data if someone asks for it. An example of the query-driven system is the sensor node, which stores data in itself and does not forward the data except upon request.

The query-driven system was used in the data gathering prototype system presented in Publication I, but subsequently the systems most often used in data gathering prototypes were time-driven (e.g., Publication II) or event-driven (e.g., Publication IV).

Rapid prototyping embedded SW/HW systems is important because system differences have increased and the product includes variation in software and system features (Buchenrieder, 2000). In addition, involving users in the specification process is crucial because an increasing number of customers expect solutions and services customized to their specific requirements. In a study, Kruger, Abu-Mahfouz and Hancke (2015) developed a working prototype using commercial off-the-shelf components. This also showed that a lengthy product development life cycle is not required when using a rapid prototyping process (Publication IV). The development of SW/HW systems is accelerated if the framework of the components used in the design of prototypes is defined. (Kreiner et al. (2001), Saha, Mitra and Basu (1997), Srivastava and Brodersen (1991), Saari, Rantanen and Hyrynsalmi (2020)

The off-the-shelf software component development and maintenance process is widely handled in the framework cited in Mäntyniemi, Pikkarainen and Taulavuori (2004). A strict focus on the applications in a software-hardware combination can also be found. For example, Rojas and Barrett (2017) introduce a platform for machine and structural monitoring.

There is also another, a wider way to divide SW/HW components - IoT architecture layers. In this, the components can be divided into layers: sensing layer, networking layer, service layer, and interface layer (Xu, He and Li, 2014), (Vakaloudis and O'Leary, 2019). In this thesis (Chapter 4), the SW/HW framework is focused on the sensing layers. The networking layer exists and is needed for data transfer, but is not the focus of our research. The interface layer is described to the user in the SW/HW framework but is excluded from the study.

The related studies introduce the design factors used through which the guidelines of development are selected. Furthermore, this section confirms the usage of selected devices in academic prototype research. The devices support rapid software prototyping with good documentation of interfaces and supported features. With the selected devices, the focus of research can be placed more on software development instead of hardware development.

Table 2.1 Programming languages used in the context of this thesis.

Language	Context
Assembly / Assembler language (ASM)	Low-level programming language. Usable in microcontroller programming.(Barnett, O'Cull and Cox, 2003)
C Programming language	"General-purpose programming language with features economy of expression, modern flow control and data structures, and a rich set of operators." (Kernighan and Ritchie, 1978)
C++	"General-purpose programming language with a bias toward systems programming". (Stroustrup, 2013)
Python	High-level general-purpose programming language. (Python Software Foundation, 2021)
(Bash) Shell scripting	Enables the task automation in Linux OS.(Nemeth, Snyder and Hein, 2002)
Java	Generic, Object-oriented programming language (Oracle, 2021)

2.4 Programming languages and hardware for prototyping WSN applications

IoT prototype systems can be built using several technologies. This section introduces the hardware and software used during the research work of the thesis.

Embedded Systems - "A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function." (Barr, 2020) For this thesis, the term "embedded system" is not suitable when talking about IoT or WSN devices and systems. IoT is an abstract term and the WSN is more precisely defined. Therefore, the pair of terms “Software” and “Hardware” are used in this thesis. The studies focused on WSN solutions where the software and hardware are usually handled together and it is hard to find research where one part is not mentioned at all.

Single Board Computers (SBC) are commonly used technology in the development of prototypes (Saari, Baharudin and Hyrynsalmi, 2017). Developing a prototype is often deemed challenging and expensive, primarily due to the costs involved

in hardware and software design, development, and manufacturing. However, employing SBCs can mitigate these expenses. There are already off-the-shelf hardware solutions, like the Raspberry Pi, with pre-installed embedded Linux software. Additionally, numerous online communities and user groups are available to offer help and support to developers. (Saari, Baharudin and Hyrynsalmi, 2017)

This thesis focuses strongly on prototyping and not on reliability issues. Therefore, for example the hardware used is not designed to operate in an industrial environment.

The programming languages used during the thesis work are listed in Table 2.1. In the early years of this research (2000-2005), the microcontrollers were programmed with C programming language which gave the possibility to modify the compiled assembly code. The focus of the modification was to improve the assembly code and software produced by the compilers before it was uploaded to the microcontroller chip. Nowadays (2023) the programming tools used are similar to Arduino IDE[1] with SBM. This tool enables the use of C and C++ type programming languages.

Shell scripting and Python are used to control the operation of the system. Shell scripting enables simple automation such as opening the data connection, file transfer, and log file writing. The Python programming language has more capabilities thanks to its support libraries[2]. These libraries can be used for connection and data transfer to third party software, such as database systems. In addition, Python has good capabilities for data manipulation if needed (Perkovic, 2012).

In the thesis context, Java is a more specific programming language compared with the above. Java is used in the Android based smartphone environment and there is no need to use other languages (also the User Interface (UI) needs XML and the compiler needs configuration files).

Hardware in the context of this thesis usually means the WSN sensor node, which collects the data in some way. A basic sensor node consists of the sensor, CPU, communication module, and power supply (Healy, Newe and Lewis, 2008). The study by Ojo et al. (2018) divides IoT devices, with a lot of examples of hardware, into three categories: low-end, middle-end, and high-end. The low-end devices are the most restricted, with small processing power and a small amount of Random Access Memory (RAM). The data gathering prototypes presented in this thesis, for example Arduino, are in this device category. Middle-end IoT devices have more

[1]https://www.arduino.cc/en/software
[2]https://docs.python.org/3/library/

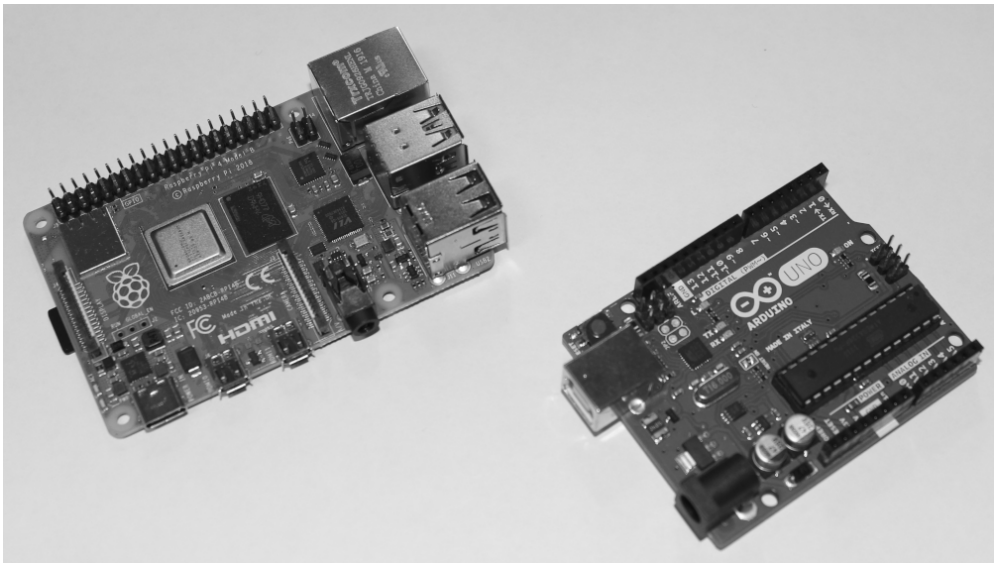

Figure 2.4 SBC Raspberry Pi and SBM Arduino Uno.

processing capabilities and memory. In the last category, high-end IoT devices, SBCs have enough capabilities to run OSs like Linux OS. The data gathering prototypes in this thesis use this level of devices for applications where the data are more complex or the amount of data is larger.

The hardware and devices used in our data gathering prototypes are mostly off-the-shelf devices, which means that the devices are available in electronics stores at a reasonable price. More reasons to use off-the-shelf devices are as follows:

- cheap to buy
- easy to use, good documentation
- community support
- widely used
- widely configurable

These features are most often the advantages of prototypes, but it should be noted that, for example, "community support" may mean a single enthusiast somewhere in the world. This kind of research problem arose when the commonly available NB-IoT development board was used in the study by Rantanen et al. (2021).

During the thesis research, several different devices were used when constructing data gathering IoT prototypes. Two main hardware devices were the Raspberry Pi

SBC [3] and Arduino Uno SBM [4] (Figure 2.4). The third experiment device family was Android smartphones, which were used as is without hardware modification. The smartphone includes the basic features of an IoT sensor device as a ready-made package: power source, processing power, various communication capabilities and modifiable software. The smartphone experiments were application layer experiments on top of the smartphone OS. The software was developed to use the phone's capabilities, e.g., sensors. Table 2.2 presents the most useful features of different hardware in the thesis context.

Several different SBC devices were used to make the prototypes, as shown in Table 2.2. The Intel Galileo (Intel Corporation, 2014) and BeagleBone (Coley, 2014) devices are comparable to Raspberry Pi and all of them are low-cost development devices suitable for testing or educational purposes. The Raspberry Pi is the most popular of the three in the field of research according to the keyword search in the IEEExplore database (December 21, 2020: "Raspberry Pi" - 3317 hits, "Intel Galileo" - 58 hits, "BeagleBone" - 133 hits).

2.5 Data gathering prototype development process

The IoT prototyping process can be viewed from two perspectives: a software development process and an embedded hardware development process. Furthermore, the author researched the IoT data gathering prototype development process by collecting data from several prototyping processes.

Regarding software related development processes, three different ways to model a development process are presented in Figure 2.5. The "Waterfall" model (Royce, 1970) represents the steps in developing large computer programs, but it has limitations, for instance returning to an earlier phase is forbidden. The second model in Figure 2.5 – Agile software development process model allows a cycle of different phases. Agile software development is the use of light but sufficient rules of the project behavior and the use of human- and communication-oriented rules (Cockburn, 2007). Furthermore, Agile processes value code production more than plan-driven processes (D. Mishra and A. Mishra, 2011). The last model in Figure 2.5 is rapid prototyping, which is based on rapid development cycles.

[3] https://www.raspberrypi.org/
[4] https://www.arduino.cc

Table 2.2 Features of Arduino Uno and Raspberry Pi hardware.

	Raspberry Pi	**Arduino Uno**	**Android Smartphone**
Price (Euro)	35	20	50-1000
Release date	24 February, 2012	20 February, 2010	OS Version 2.3, 9 February 2011 (Nexus S was our first developer enabled smartphone)
Recent version	Raspberry Pi 4 B	Arduino Uno Rev3	OS version 11 (version 10: Nokia 8 and 7.2 devices)
Processor	24 Broadcom BCM2711B0	ATmega328P	Qualcomm Snapdragon 660
Memory	2- 8 GiB	32 KB, 2 KB, 1 KB (flash, SRAM, EEPROM)	4/6 GB RAM
OS	Suitable for e.g., Linux, OpenBSD, Windows 10 ARM64	None	Android
References	Raspberry Pi Foundation (2020)	Arduino (2020)	*Android (operating system)* (2021), HMD Global (2021)

Liou (2019) presents the basic ideas of prototyping in his book "Rapid Prototyping and Engineering Applications". Rapid prototyping can be represented as a circle (Figure 2.5). Rapid prototyping includes three stages: making a prototype, reviewing the result, and refining and iterating (Babich, 2019).

Both rapid prototyping and Agile require the making of incremental improvements over several iterations, but implementing a prototype focuses on rapid prototyping, while Agile's focus is on the product. The idea of rapid prototyping was used in the prototype projects of the thesis because coding guidelines could be ignored. According to Hunt and Thomas (2000)(p. 54), prototyping allows the ignoring of:

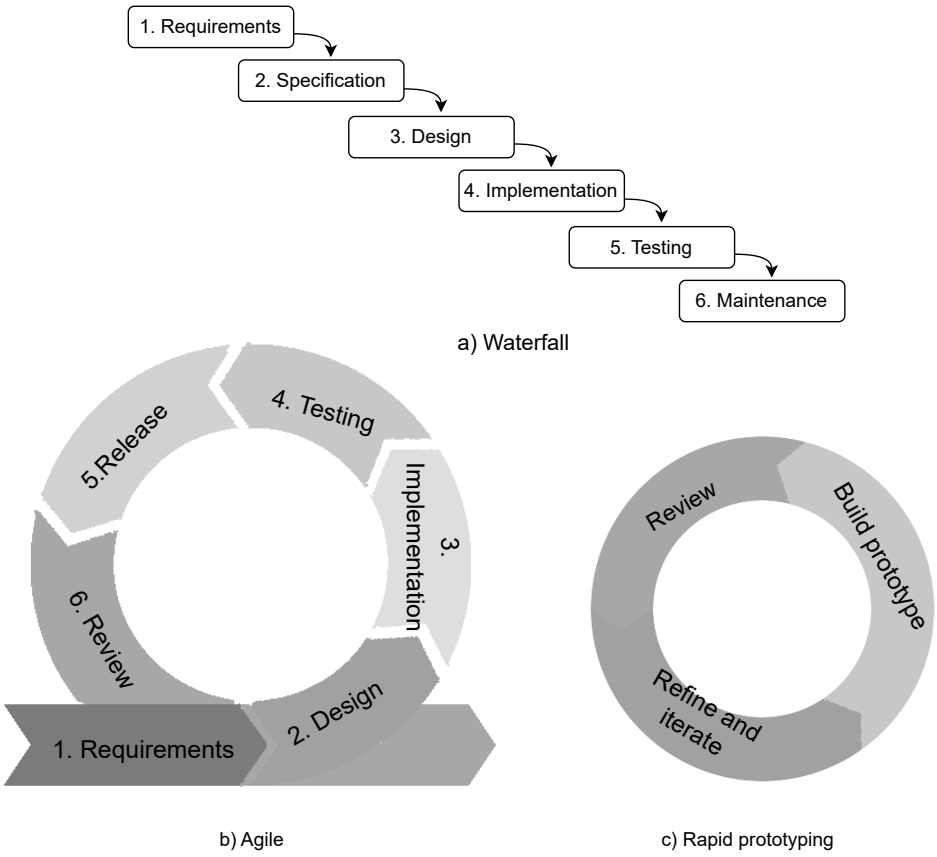

Figure 2.5 Software development process models. a) Waterfall process model, b) Agile process mode, and c) Rapid prototyping process model.

- Correctness – using dummy data
- Completeness – the prototype can work only in a limited area (i.e., input data or functionality).
- Robustness – error checking
- Style – coding of the prototype can be without style guide and documentation and comments can be missing.

In the context of IoT, the working prototype solution is the following: hardware to run the software; software for collecting, storing, and transferring data; the right technologies for use cases to make things easier for both developers and users. (Publication VI)

During the research phase of the thesis, several prototypes were developed and these include a lot of software development. Therefore, a suitable development process from the area of software development was selected. For this research, there are two significant process models: prescriptive and descriptive. The prescriptive model describes how the process should be performed. In the software development context the "Waterfall" model (Royce, 1970) is an example of the prescriptive process model (Scacchi, 2002). A descriptive model describes how a process is performed in a particular environment (Becker, Hamann and Verlage, 1997).

The descriptive process model (DPM) (Becker, Hamann and Verlage, 1997), (Becker-Kornstaedt and Webby, 1999) introduces an eight-step approach to producing a process model from software processes. These steps are divided into two phases: the setup phase and the execution phase. The eight-step approach was used when the descriptive model for the prototyping process (DMPP) was developed, as presented in Chapter 5 .

Furthermore, regarding embedded systems there is some need for focusing the development process. The book "Introduction to Embedded Systems – A Cyber-Physical Systems Approach" by Lee and Seshia (2017) is based on the idea that designing and implementing an embedded system consists of three major parts of the process: modeling, design, and analysis. (Lee and Seshia (2017), Publication IV)

2.6 Summary

This section gave an overview of the thesis within the area of the research questions presented in Section 1.2. The first section, 2.1, clarified the research area by introducing the related standards. In summary, all three standards are worth knowing and they all give valuable background knowledge about the terms and concepts. However, they are also strongly focused on stakeholder concerns and business issues, which are not the main focus of this thesis. Furthermore, these standards present high-level architecture models and frameworks, which are good guidelines, but cannot directly be implemented in the technical construction of IoT systems.

The state of the art on data gathering prototypes was presented in Section 2.3 with several related studies. Section 2.4 continued the briefing by introducing the technological terms with components from the software and hardware areas. The last section 2.5 clarifies the background of the development process models when

building data gathering prototypes.

Each topic has been addressed with sufficient precision to give the reader an understanding of the subjects of the thesis. With this background knowledge, the first main subject of the thesis can be addressed: the WSN data gathering sensor node architecture model.

3 ARCHITECTURE MODEL FOR SENSOR NODES IN DATA GATHERING

The research goal covered in this chapter is the presentation of a Wireless Sensor Network (WSN) sensor node architecture model for data gathering. The architecture model contains software and hardware components, and their interconnections. With the use of this fundamentally simple model, it is possible to create highly practical and interoperable sensor applications to gather data on environmental conditions.

The research started as a practical approach – what is possible and what is not. In Publication I, the aim was to research and solve the idea of how condition changes in indoor spaces could be observed and how to collect these data. Our research group had suitable devices and a lot of knowledge about software development, but not specifically regarding the building of IoT device prototypes. Therefore, the development work started by experimenting with the structures of different data collection prototypes. These experiments showed the need to define the guidelines for interconnecting components. The purpose of the guidelines was "to keep the architects of the system from drifting off into the blue with unimplementable or costly specifications" (Brooks, 1995), p.43.

Later on, the main research method for developing the model was iterative development. The system development research process is illustrated in Figure 1.4 and more widely in Section 1.4. In addition, the design factors of WSN were followed, where applicable (introduced in Section 2.3 and based on Akyildiz, Sankarasubramaniam and Cayirci (2002)). Design factors were used as a guideline in approaching the WSN sensor node architecture model by designing concept prototype systems.

The prototype systems were built for different purposes to gather, store, and deliver data to the user. From these prototypes, an abstract architectural model for sensor applications was built. The model was applied using the iterative develop-

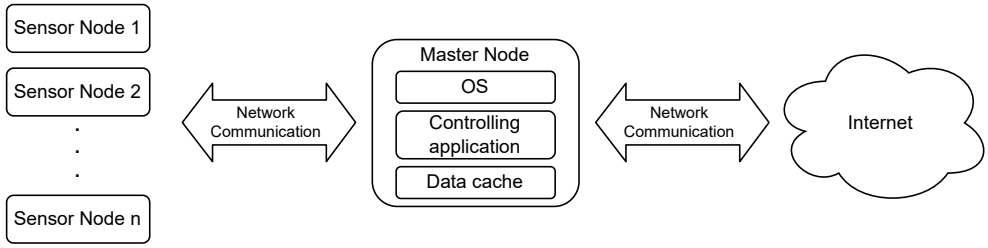

Figure 3.1 Multi node architecture model abstraction when collecting simple data on multiple sensor nodes.

ment approach and evaluated several times in the developed prototype systems. The practical outcome of this chapter is a guideline for constructing the architecture for IoT sensor applications. Further, this section creates the basis of the later introduction of the SW/HW Framework and the prototyping research process model.

This chapter starts with the introduction of two architecture models. These are evaluated and used within the IoT data collecting prototype systems, which are also presented. The last part summarizes the usability of the architecture models.

3.1 Architecture models for WSN data gathering

This section presents two architecture models developed for data gathering. The purpose of these models was to facilitate the initial design of the data collection system. The development and timeline of research produced two architecture models for different purposes. Figure 3.1 presents the **multi node** architecture model and Figure 3.2 presents the **single node** architecture model.

Both architecture models contain the following main abstract components:

- Sensor nodes, containing sensors to measure environmental conditions, e.g., temperature, humidity, acceleration, photos.
- Master node OS and controlling application, controlling the collection and delivery of the data. A data cache is useful if data analysis or processing is needed.
- Communication between sensor nodes and master nodes, and the Internet is needed.
- The Internet is the channel used when data are provided to a user or service

platform.

Multi node architecture is focused on systems that collect data from several points in an area - the communication technology chosen determines the extent of the area. In the implementations described in this thesis, a group of sensor nodes are suitable for collecting simple numerical data, e.g., temperature or humidity. The master node controls the sensor nodes and collects the data. The data can be stored, analyzed, and processed, and sent further on or offered via the Internet. The advantages of this architecture model are the ability to collect a large amount of environmental data with low power and reasonably priced devices. The limitation of the model is that the type of data to be collected is limited to simple sensor data. For example, a surveillance camera sensor network would require too much processing power from a master node and the reasonable price advantage would be lost.

Single node models are focused on systems for a smaller amount of sensors, but the collected data can be more diverse, such as photos or a large amount of numerical data chunks (e.g., the mean value of acceleration). The sensor node and master node are software components which work in a single device, such as a smartphone or Raspberry Pi with sensors. This architecture model has been designed to collect data which are at least partially processed in a device. The advantages of this architecture model are its configurable and reasonably priced devices. The limitations, e.g., data processing ability, guides device selection for more expensive devices.

The **multi node** and **single node** models were developed to clarify the inner architecture of the sensor node in the data gathering prototype systems. These models can be used as a guideline when designing IoT systems. The information collected determines which model it is more sensible to use. The architecture models do not limit the amount of systems, for example two single node systems (Raspberry Pi constructions) are suitable for collecting data from multiple locations (Grönman et al., 2019).

3.2 Developing the architecture models with prototype systems

This section follows the timeline of data collection model development. Publication I presents the starting point for the research – the first prototype and the background

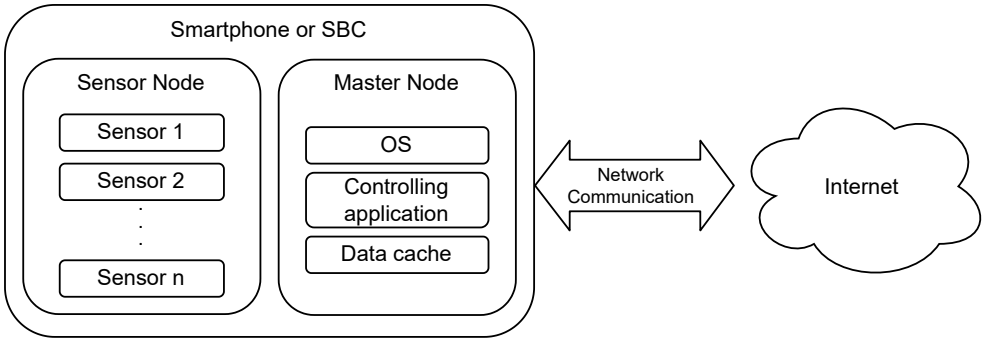

Figure 3.2 Single node architecture model abstraction when collecting larger data from a single location.

of the development process. This is followed by the second prototype system and the **multi node** architecture model of WSN data gathering. The model defines the components to build the working prototype at an abstract level. The **single node** architecture model is introduced in Publication III. In this further developed architecture model of data gathering, the hardware configuration is replaced by a smartphone, a fully operational embedded off-the-shelf sensing device.

The **first prototype** system and basis of the study was the intelligent alarm system that was under development at Keio University Shonan Fujisawa campus[1]. The intelligent alarm system consists of several data gathering prototypes – Data Collector Services. (Publication I)

Publication I presents the first prototype system (Figure 3.3) that was developed. This is an early phase **single node** architecture construction. The hardware consists of a BeagleBone Black SBC, which runs on Linux OS (Ångström) (Coley, 2014). In the OS, database services were constructed to store data and a web service to offer the data to the Internet. BeagleBone has an Ethernet connection to the Internet. The data were collected with sensors: the first sensor collected humidity and temperature data and the second one, a photo-conductive cell, collected brightness data.

In conclusion, Publication I presents the features and architecture, the hardware and software components of the prototype, and the physical connections between the components. The structure of the developed software used for data collection is also described.

The **second prototype**, a **multi node** system developed for WSN data gathering contains a master node managing several sensor nodes. The architecture is presented

[1]https://www.sfc.keio.ac.jp/en/

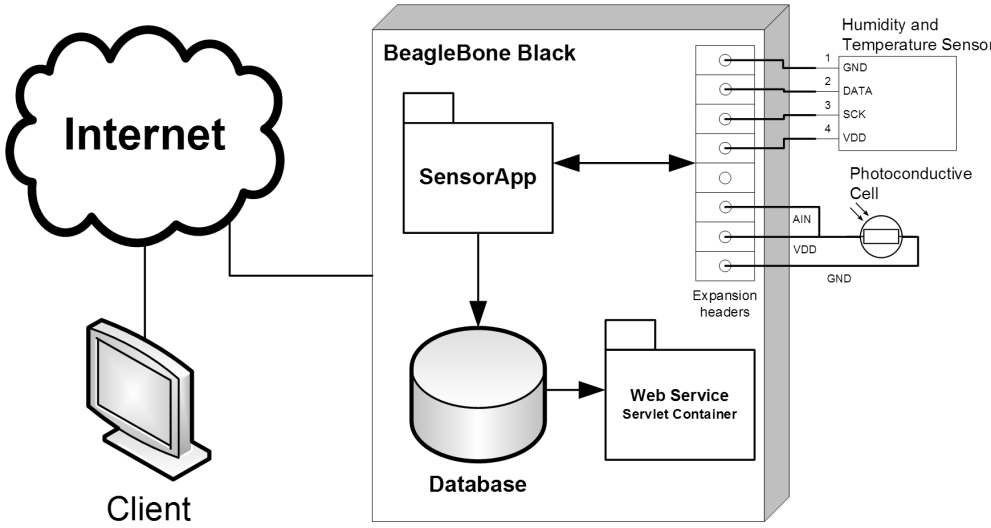

Figure 3.3 System Architecture of Data Collector Service - Starting point of data gathering model. The figure is adapted from Publication I.

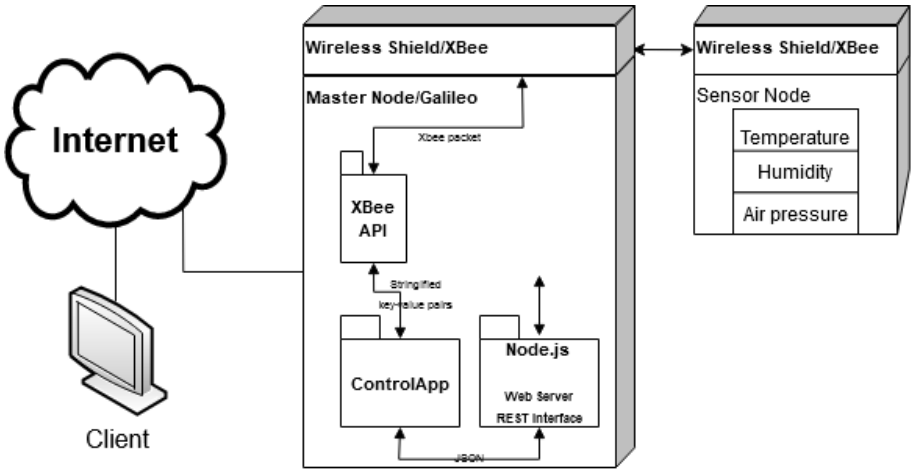

Figure 3.4 System architecture of the prototype system introduced in Publication II. In a **multi node** system, the master node gathers sensory data from several sensor nodes.

in Figure 3.4. The sensor nodes collect data from several points and route the data to one master node. The collected data are stored on the master node and are provided to the user via the Internet. (Publication II)

The basic features of the second prototype (Publication II) are as follows:

- Collecting temperature, humidity, and air pressure data from the environment

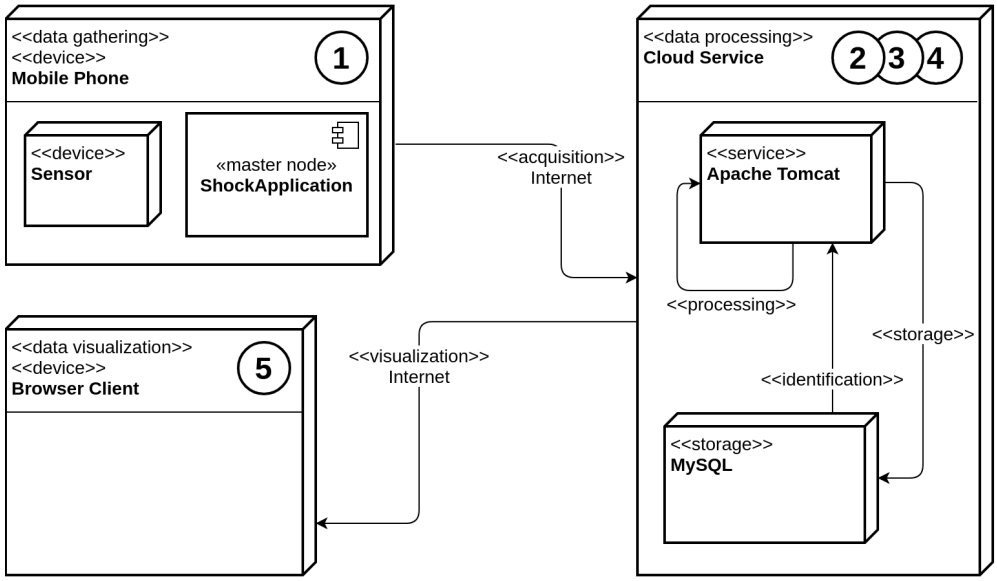

Figure 3.5 System deployment diagram for the "ShockApp" prototype system. The figure is adapted from Publication III.

with several sensor nodes. The number of sensor nodes is not limited.

- Data collected and cached on master node SBCs: OS in SBC offers the database application.
- Data processing: no processing.
- SBCs use an Ethernet connection.
- User access to data: Data provided to the clients over the Internet.

The **single node** architecture model was introduced in a data gathering pilot study by Sillberg et al. (2018), where the focus was on collecting data from several sensor nodes. The developed prototype system and architecture model are presented in Publication III (Figure 3.2. Figure 3.5 presents the deployment diagram of the developed data gathering prototype system. The mobile phone application collects data from the mobile phone sensors, the data are stored on the phone and then sent to the cloud service. The users and client can fetch or process the data in the cloud service. The test software named ShockApp was developed during the study. The application and its features are discussed in depth in Sillberg et al. (2018). (Publication III)

The usage of smartphones enables the crowdsourcing idea, where a large amount

of people use the software and collect data. ShockApp can be installed on all modern Android smartphones. The identification mark of the user helps to order the data points in the cloud. The data are cached on the smartphone and stored in a cloud service. Publication III uses (Figure 3.5) the relational database MySQL. The other possibility is to use a time series database, as presented in the accompanying research study Saari et al. (2020a). The cloud service enables the processing and usage of the data.

3.3 Evaluating architecture models with prototype systems

The architecture models were evaluated by building prototype systems. The topic of prototype testing methods and procedures was oriented by means of a literature survey entitled "Survey of Prototyping Solutions Utilizing Raspberry Pi" (Saari, Baharudin and Hyrynsalmi, 2017), although the focus of the study was more about the usage of Raspberry Pi for prototyping purposes than finding overall prototype testing methods. The survey emphasized the limited use of testing practices and methods in this context. Often only functional testing was performed, which in practice means testing that the prototype works. In our prototype systems, the testing was also functional testing, i.e., does it work as planned? In some cases, a service stress test or data transfer coverage test was also performed.

The **multi node** architecture model was evaluated with off-the-shelf SBCs and other instruments in three studies: Publication I, Publication II, and Baharudin et al. (2016).

Publication I presents a data collector service that utilizes a BeagleBone Black computer and sensors. The goal of the study was to evaluate how an off-the-shelf SBC could be used to collect sensory data and how this data could be provided to the client over the Internet. The developed prototype produces data regularly and has proven to be stable and reliable in practice. The research presents the features and architecture of the developed service, the hardware and software components used, the physical connections between the components, and the structure of the software. The research gives a concrete example of using an SBC with embedded Linux distribution. In addition, the study presents the design of the system, which was tested and found to work as planned. (Publication I)

Publication II presents a WSN implementation of a prototype system. The em-

Table 3.1 Design factors and how they are implemented in the prototypes. Publications I and III describe the single node system and Publication II describes the multi node system.

Design Factor	Implementation
Fault tolerance	Publications I and II - Automatic recovery from power outage managed by OS. Publication III highlights OS version related software problems.
Scalability	Publications I-III introduce independent software based constructions where the amount of devices was unlimited. Publication II describes the separate sensor node-master node construction. There is a theoretical limit for the amount of sensor nodes within a master node, but this was not reached in the tests.
Production costs	Off-the-shelf hardware and Open source or free-to-use software. The hardware constructions and software development (or configuration) were made by the research group.
Operating environment	Indoor and outdoor usage tested. Further, the environment for using the prototypes is not limited.
Sensor network topology	Star topology or its variants. The special case of fog gateways is handled by Baharudin et al. (2018)
Hardware constraints	Publications I and II use SBCs - Sensing unit (sensors), data storing and processing unit (Linux OS), and transceiver unit (network communication). Publication III introduces the prototype implemented in a mobile phone (Android smartphone). Only software modifications made.
Transmission media	Publication I - Ethernet network. Publication II - Ethernet and ZigBee communication. Publication III - Mobile network(3G, 4G)
Power consumption	Baharudin et al. (2016) deals with maximizing energy savings through software-related optimization.

Table 3.2 Summary of benefits and challenges of single node and multi node architecture models in prototyping for the design factors presented in Table 3.1

	Single node e.g., SBC prototype or smartphone	**Multi node** e.g., SBC master node with several SBM sensor nodes
Fault tolerance	Allows easy booting recovery from a fault condition. Difficult to start remotely.	Easy booting – the master node will restore the fault state. Sensor nodes are difficult to boot remotely if configured without remote access (only send data).
Scalability: Number of physical devices	Scales linearly with amount of devices.	Single master node has theoretical limit of sensor nodes, but it not reached in prototype systems.
Scalability: Data	Enables complex or large size data packets.	Restricted for small size data packets.
Production costs	Consumer product smartphones can be used. SBC production costs with off-the-shelf devices are low.	Price depends on the quality and quantity of sensor nodes.
Sensor network topology	Allows multiple network topologies (star network tested).	Allows multiple network topologies (star network tested).
Transmission media	Allows use of multiple network communication technologies. Smartphone has internet connectivity built in.	Possible to optimize the range, power consumption, or price when selecting the sensor node communication technology. Allows the use of multiple network communication technologies.
Power consumption	Moderate power consumption with SBC; smartphone has limited battery operation allowing short power blackouts.	Allows sensor nodes to work with low power consumption (depends on selected software / hardware).

bedded Linux controlled sensor network utilizes sensors for the Arduino SBM sensor nodes and an Intel Galileo SBC for the master node. ZigBee expansion boards handle the communication between sensor nodes and master nodes. In the protocol stack related to ZigBee, the Physical Layer(PHY) and MAC (Media Access Control) Layer are below the ZigBee network layer, and the Application Layer is above it. The study proved by testing that cost-efficient SBCs have the ability to gather data from sensor nodes and provide it to users over the Internet. Publication II also presents the features and architecture of the developed service, the hardware and software components used, the physical connections between the components, and the structure of the software. The research gives a concrete example of using an SBC with embedded Linux distribution. (Publication II)

The third study, (Baharudin et al., 2016) used a multi node architecture model when testing a low-energy algorithm for data transfer between the sensor nodes and the master node. The system architecture of the prototype was similar to that presented in Figure 3.4. The master node receives data from the sensor nodes via ZigBee communication. The focus of the study was on reducing energy consumption due to wireless data transmission without sacrificing the reliability of data for real-time visualization. The total power consumption can be significantly reduced by applying the algorithm to the sensor nodes.(Baharudin et al., 2016)

The **single node** architecture model is presented in Sillberg et al. (2018) with a description of the ShockApp application in smartphones. Furthermore, Publication III extends the study by Sillberg et al. (2018) by introducing a combination of models for data gathering and an analysis of the collected data.

The deployment diagram of a system that utilizes data collected by smartphone sensors is presented in Figure 3.5. The data were obtained from a group of smartphone users driving on the roads in western Finland. The developed smartphone testing software itself is an Android application, which consists of a single main view that allows the user to stop and start the sensors, and a settings screen, which allows, for example, the modifying of user credentials. The application collects accelerometer data, direction, speed, location, and timestamps. The data are stored in a cloud service at predefined intervals. (Publication III)

Publication III combines a data gathering model with a data analysis model to evaluate the condition of road surfaces. This was implemented and tested in a prototype system. The trial period showed that the selected methods were successful in

identifying individual road surface issues, such as potholes, but more importantly, they were also effective in providing an overall assessment of the road condition. (Publication III)

Despite the variety of hardware used, the prototype systems presented in Publication I and Publication III use the architecture model where master node and sensor node are separated into separate software components. The separation is implemented at software level, creating the necessary software components.

The **implementations** of the different evaluations are listed in Table 3.1. The table summarizes the practical implementations, comparing the design factors presented in 2.3. These practical implementations are also categorized by the design factors (Akyildiz, Sankarasubramaniam and Cayirci, 2002).

The **benefits and challenges** of single node and multi node architecture models in prototyping are listed in Table 3.2.

3.4 Discussion and summary

The goal of this chapter was to answer the first research sub-question: **RQ1:** What kind of sensor model architecture can be developed for data gathering in a wireless sensor network? The design factors from Akyildiz, Sankarasubramaniam and Cayirci (2002) served as a guideline for prototype development. The study by Barrenetxea et al. (2008) presented time-driven, event-driven, and query-driven categories for WSN systems. The architecture of the sensor node model can work in all of these categories. The first prototype introduced in Publication I is a combination of time-driven and query-driven, where the data are collected periodically and served only in a query. As explained in Chapter 4, two event-driven prototypes are presented in Publication IV.

The main contribution of this chapter and the first artifacts of the research are the **multi node** and **single node** architecture models. The chapter described the main components: the sensor node, master node, communication, and the Internet as a communication channel for user applications. Furthermore, the purpose of the models and possible applications as a data collection tool were presented.

The architecture model and prototype development along with the related prototypes were described in Section 3.2. The evaluation of the architecture models was approached by constructing prototypes. The working prototypes show the usabil-

ity of the architecture models as a guideline when constructing data gathering sensor devices.

The research work on this chapter was done by means of several prototype systems. The most significant findings and results from the prototype systems are presented in Publications I-III and this chapter presents them as a single entity.

Publication I shows the ability to prototype data collection with reasonably priced off-the-shelf devices. The data collection, analysis, storage, and offering were demonstrated with the proof-of-concept prototype. Furthermore, the prototype system highlights the strength of component based prototype development where some components are ready to use with only slight configuration or modification. Publication I presents the first implementation of the **single node** architecture model.

Publication II presents the proof-of-concept implementation of the **multi node** architecture model which was utilized to construct sensor nodes – master controller combinations in the IoT environment. The evaluation of the **multi node** model shows the usability of an off-the-shelf SBC to collect data from multiple sensor nodes.

Publication III demonstrates the ability to use lessons learned from earlier **multi node** prototypes with a new device family of smartphones. The presented proof-of-concept ShockApp smartphone application showed the usability of smartphones in an IoT data gathering system. The same component based ideology as described in Publications I and II was used but only the software side; the system used consumer devices without hardware modifications. Therefore, the smartphone application is a **single node** architecture model implementation.

The related research section 2.3 also introduced several studies related to the modeling topic. Regarding the thesis, the node level abstraction, which is also the idea and starting point of Chapters 3 and 4, is presented in Laukkarinen (2015).

Despite the similarity of the idea, more ready-made off-the-shelf components were used in the development of prototypes. Building the prototype systems showed that the hardware selection supports rapid prototyping and that test configurations were reproducible. Furthermore, several related studies were selected for constructing a similar off-the-shelf component tool set. These studies showed that the hardware used, i.e., Raspberry Pi, Arduino, and similar devices, were suitable (and commonly used) in the academic research area for prototyping and evaluating ideas (Saari, Baharudin and Hyrynsalmi, 2017).

This chapter presented the basic approach when building the first implementa-

tions of the architecture of data gathering prototype systems.

Table 3.2 lists the benefits and challenges of the architecture models. The abstraction of the models is the main feature, so developers have ample opportunities to make choices between different features and different technologies. Furthermore, choices made by developers can significantly affect the price of the prototype, making cost-effective development possible if desired.

The presented two models can handle wide variations of data gathering prototype systems, which is proved by the following Chapter 4. The presented architecture models are focused on the data gathering side of systems and therefore they do not deal with data storage, usage, and processing on the cloud side. Future research topics could focus on the cloud side, closer to data use and utilization.

The results described in this chapter give a strong background to the development of the framework presented in the next chapter. After formulating the architectural model, the research continued with the development of a framework that utilizes the model.

4 FRAMEWORK FOR IOT PROTOTYPE DEVELOPMENT

This chapter introduces the second artifact of the research – the Software / Hardware framework (SW/HW framework) for IoT data gathering. The framework generalizes prototype development into a group of required components; more specifically, the framework defines the guidelines for constructing prototype systems to collect data for different purposes. In addition, this chapter answers the research question **RQ2:** How can IoT data gathering be generalized into a framework of required software and hardware components?

Development work on the framework began from the previously created **multi node** and **single node** architecture models. The research method used in this phase of the study was the same as that introduced in 1.4, i.e., data were collected when the prototype artifacts were implemented, and these data were used to develop the framework. This chapter is based on Publications IV-VI.

The SW/HW framework is a guideline for producing practical implementations – data gathering IoT prototypes. Three different constructions are presented for building prototype systems. The software and hardware components are presented for all three types of construction. Also, the practical implications are introduced where these constructions have been evaluated.

This section starts by presenting the research approach, the timeline of the research, and the studies that are relevant to the SW/HW framework development. The SW/HW framework is presented by introducing the sensor network environment, three types of data gathering constructs, and the components with their interconnections. Section 4.3 introduces the use cases that are congruent with the construction of framework Types 1-3. Also, some examples of program code are presented from selected real-life use cases to illustrate the role of the different components in the system. Finally, the benefits of using this developed framework are

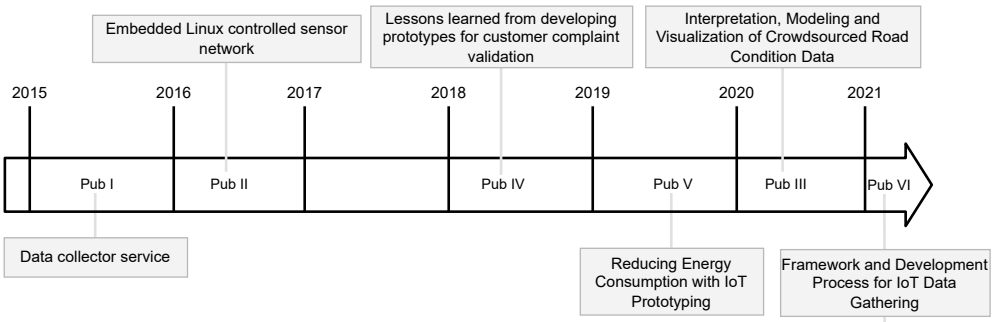

Figure 4.1 SW/HW framework validation with prototypes in the timeline of the thesis research.

listed.

4.1 Development of the SW / HW framework

The SW/HW framework was produced from several data gathering prototype systems, which have been developed and evaluated during several research projects. Figure 4.1 presents the publications in a timeline. Publications I and II describe the start of the prototype development with the first experiments of suitable software and hardware. The study by Saari, Baharudin and Hyrynsalmi (2017) explores more broadly what others in the academic world have done within this research area, identifying the lack of formalized testing methods and minimized testing in general when developing prototypes for research purposes.

Five different prototype systems for data gathering are presented in four studies: Sillberg et al. (2018), Publication IV, Grönman et al. (2019), and Rantanen and Saari (2020). The main idea was to design a data gathering system and test it with suitable off-the-shelf devices and open source or free to use software. One of the studies, Sillberg et al. (2018), was expanded into Publication III including modeling aspects and was not made public until 2020.

Publication IV can be said to be the first iteration of the SW/HW framework. The study presents the ideas and criteria for the components that are suitable for data gathering prototypes. Publication V includes studies with potential methods and technologies for monitoring energy consumption and savings. Also, Publication V summarizes the proof-of-concept demonstrations and prototype applications that were developed to illustrate how to utilize cost-effective, open, and modular so-

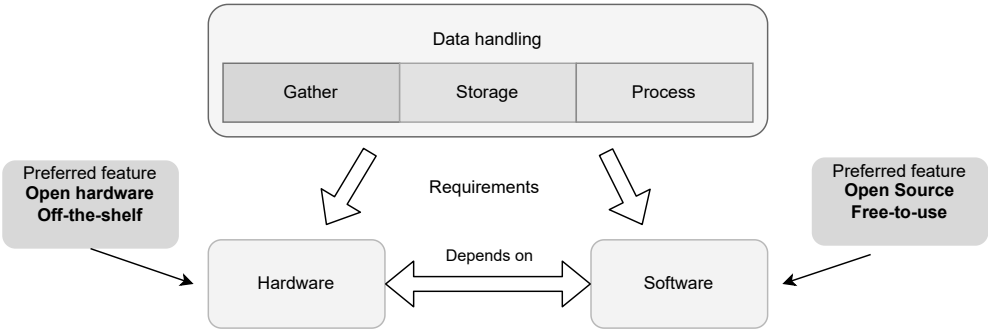

Figure 4.2 Requirements from data handling guide the selection of software and hardware in the SW/HW framework.

lutions. Saari, Rantanen and Hyrynsalmi (2020) includes the second iteration of the SW/HW framework. This work was extended to become Publication VI. Chapter 4 is based on Publication VI with minor additions.

4.2 SW/HW framework

This section introduces the SW/HW framework for an IoT data gathering system. The abstract **multi node** and **single node** architecture models were introduced in Chapter 3. The framework is the concrete application framework which uses the developed architecture models.

The application framework is a guideline for producing practical implementations of data gathering sensor prototypes. The main purpose of the SW/HW framework is to guide and assist in the construction of data gathering prototypes, and therefore a set of hardware and software components to use for building data gathering systems is introduced. The advantage of the framework is the support of reusability, portability, and interchangeability.

The SW/HW framework was defined during academic research projects where the focus was on data gathering with self-made prototype systems. The framework and its relationships are illustrated in Figure 4.2.

Essentially, when collecting data for a project, the types of data to be collected determine the hardware and software components required. Optional features that can be helpful can also guide the selection process. Ultimately, it is important to ensure that the chosen hardware and software work together seamlessly.

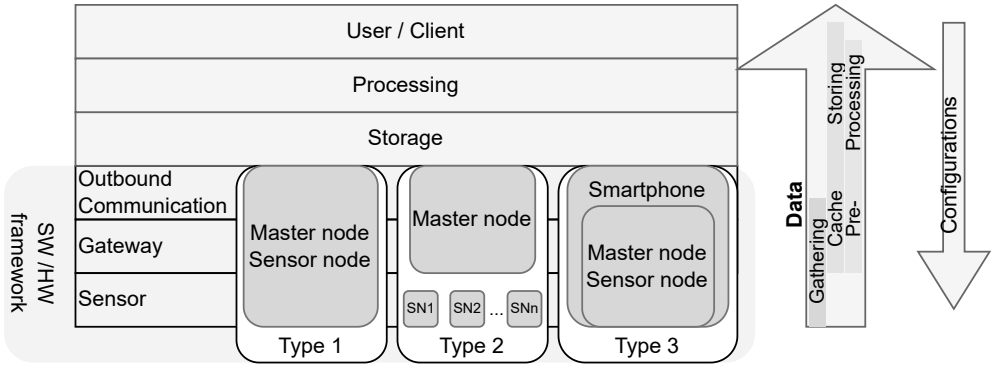

Figure 4.3 The SW/HW framework within the layer diagram for gathering IoT data.

"What information or data to collect" is the first question for the implementer of an IoT data gathering system when constructing a prototype. The answer to the question should be clear and it should also be the motivation for constructing the system. During the thesis research work and the construction of prototypes, the following three questions were asked after the decision of collecting data had been made (Publication VI):

- How to gather the data?
- How to store the data?
- How to process the data?

The availability of the collected data is crucial for the end user. To ensure this, it is preferable to store and process the data in cloud storage or a server, which could be an old Linux server or a commercial cloud computing service. However, before sending the data to the cloud storage, they should be collected and temporarily stored in a sensor device or gateway device. This ensures that if there are any communication issues with the network, the data can be saved temporarily, reducing the risk of data loss. In this context, the SW/HW framework is designed to focus on data storage and processing at the sensor or gateway layers. This allows for more efficient data management and helps ensure that the end user has access to the data when needed. (Publication VI)

Figure 4.3 presents an IoT data collection system with layers starting from data sensor and ending with the user of the data. In this layer presentation, the SW/HW

framework, and all three types of configurations are located in the lower layers. The following parts of the system should be noted:

- The collected data are utilized by the user/client (human or computer application).
- In most use cases, the data are processed in some way according to customer needs.
- The raw data are collected and saved in the storage layer.
- The outbound communication layer offers a suitable data transfer method for the SW/HW framework. The most developed prototype systems have a connection to the Internet, but this is not mandatory.
- In the SW/HW framework, three different hardware constructions (Types 1-3) are presented: master node-sensor node; multiple sensor nodes with one master node; sensor device (for example a smartphone). Types 1 and 3 are derived from the **single node** model and Type 2 is derived from the **multi node** model.
- On the gateway level, the data could be cached and/or processed if it is necessary and possible.
- Remote control for monitoring and configuring is enabled.

The versatility of the SW/HW framework is supported by dividing the main node-sensor node into three different structures, which enables the collection of a versatile data set. (Publication VI)

Hardware of the SW/HW Framework

The **hardware** of three types of SW/HW framework constructs uses off-the-shelf devices; the most commonly used hardware is listed in Table 2.2. The use of at least partially tested ready-made devices speeds up the development of prototypes. The devices can be categorized into two parts (Publication VI):

- Sensor node hardware contains a combination of data sensors and a control device.
- Master node hardware for gathering, processing, and storing of collected data.

Table 4.1 Most notable features of the three different types of construction referred to in Chapter 3 including presented architecture models (Publication VI)

	Type 1	Type 2	Type 3
Architecture model	Single node	Multi node	Single node
Hardware construction	SBC with sensor(s), SBC works as a control device of sensor(s)	SBC master node and group of sensor nodes with sensors	Smartphone
Data gathering	No limitations – suitable for large data chunks such as photos	Suitable for low data transfer – SBC limitations	Device sensors – no hardware modifications
Data processing	SBC limitations	e.g., mean value calculus, visualizing	e.g., mean value calculus, data packaging
Data storage	Temporary storage	Temporary storage or Database storage and visualization	Temporary storage

When compared to the three types of data gathering devices, the division into three parts is sufficient for hardware. Types 1 and 3 are combined on the hardware side, with sensors and processing capabilities integrated into a single device. Type 2, on the other hand, allows the master node to manage multiple sensor nodes. Table 4.1 includes references to previously presented architecture models. (Publication VI)

The data collection process is executed by the **sensor node**, which employs sensors to gather the necessary data. The data could be basic information, like temperature and humidity, or more intricate data, such as photos. The hardware utilized is determined by the data requirements. The sensor node is separate due to being Type 2, where the aim is to use multiple sensor nodes with a single master node. Construction sensors are connected to an SBM, i.e., an Arduino or similar, in Type 2, which can manage simple data, such as numerical values. The Type 2 sensor nodes are connected to the master node, and the volume of transferred data must be in bytes or kilobytes. Android smartphones can process a significant amount of basic information from their built-in sensors, as well as more intricate data such as images

captured by the device's camera. (Publication VI)

The **master node** receives data from the sensor nodes (sensor nodes send data or the data are fetched by the master node). The master node can preprocess and/or cache data if needed. The master node has a communication channel, for example, 3G/4G/5G[1], Wi-Fi[2], or LoRa. Depending on the master node's communication channel, remote control and configuration can be enabled. For example, Raspberry Pi with Linux OS and a suitable communication channel is achievable with remote control tools. (Publication VI)

The Type 3 construction is based on the idea of a master node in a smartphone and in such cases the solution is implemented with a self-made application, which handles data collection, storage, and processing. The application limitations come from the OS of the smartphone and the fact that no hardware changes or modifications have been made. (Publication VI)

This SW/HW framework relies on communication with the Internet, but in addition, suitable wireless technologies can be used (e.g., ZigBee, LoRa, 4G). The collected data are transferred via the Internet to data storage devices. This storage could be cloud servers with a database or dedicated Linux servers for saving data. The study by Saari et al. (2020a) presents several ways for storing and visualizing sensor data. The SW/HW framework can also be applied to these techniques. (Publication VI)

Software of the SW/HW Framework

The SW/HW framework primarily relies on open-source software, which has been in most cases tested by the community and comes with freely available source code. Since open-source software is also free to use, multiple software combinations can be utilized for testing without extra costs. (Publication VI)

The software components are divided into three parts:

- Sensor software - receives sensor data from sensors
- Data gathering and preprocessing software
- Data storage software

[1]Generations of wireless mobile telecommunications technologies
[2]Wi-Fi is the IEEE 802.11 standard based family of wireless network protocols

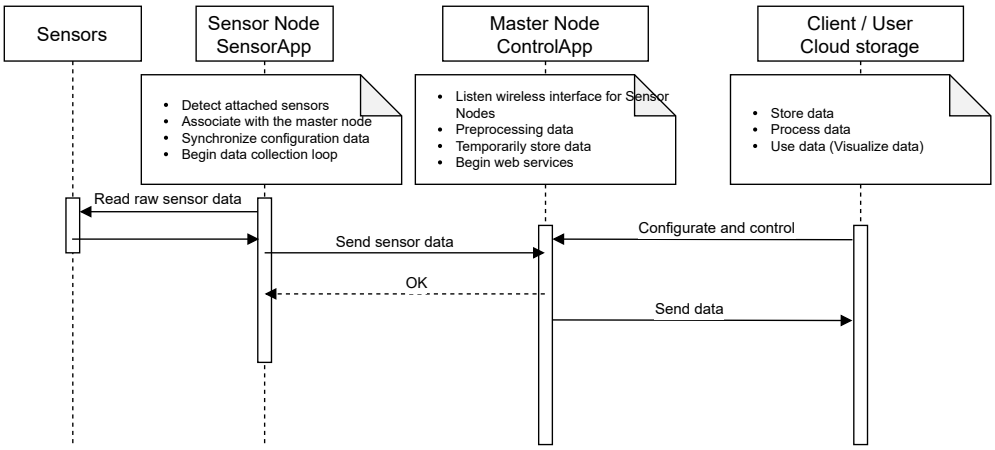

Figure 4.4 Software components and their connections when collecting data.

The diagram shown in Figure 4.4 serves as a guide for dividing software components between the master node and sensor node at an abstract level and demonstrates how these parts work together. This kind of approach facilitates modular development and the ability to interchange components. Additionally, the diagram depicts how the sensors provide input and how data are transmitted to the cloud. (Publication VI)

The pseudo code examples of the SensorApp (4.1) read the state of the sensor and send the data. The ControlApp (4.2) receives data and sends the data to the cloud. These programs illustrate the software components of the data gathering sensor node. The SensorApp (Program 4.1) works in a loop, reading the sensor measurement and sending the read value at appropriate intervals. In this example, the interval is 1000 milliseconds. The ControlApp (Program 4.2) waits for the data from the SensorApp and, depending on the setup, the data can be handled in various ways. However, the last step of the ControlApp is to send data to the cloud database. All three types of data gathering constructions implement these two programs 4.1 and 4.2; the programming language varies depending on the working environment.

```
1 setup{
2   environment configuration
3 }
4
5 loop{
6     data = read_sensor;
7     send(data);
```

```
    sleep(1000);
}
```

Program 4.1 Pseudocode for SensorApp implementation

```
void read_send_data{
  receive data
  if check==true
    check data
  if store==true
    store data locally
  if modify data==true
    modify data
  Send data
}
```

Program 4.2 Pseudocode for ControlApp implementation

Table 4.2 presents all three types and the relevant software aspects. The list of software components goes from sensing to storing the data. Low-level programming with C++, Python, or Perl scripts is suitable in Types 1 and 2 for sensing and sensor software for reading, reviewing, and storing the data. (Publication VI)

SBCs can serve the dual purpose of sensing and data gathering. In Type 1, for example, the Raspberry Pi is connected to the sensors and utilizes sensor software to read values from them. If a Type 2 construction is used, the gathering software in the SBC can handle the data collection from several SBM sensor nodes. Also, Type 2 SBC devices should be equipped with the full OS if data gathering and preprocessing software are more complex. In the data storing and preprocessing phases, the SW/HW framework utilizes pre-made software and libraries. For example, Raspberry Pi could offer the gathered data to the Internet with a server application. Preprocessing could be, for example image recognition using image recognition software. (Publication VI)

The assumption in the SW/HW framework is that the collected data are stored in a cloud server. In addition to this, the data could be temporarily saved to the master node using a suitable database format. If the data meet the definition of a time series: "A sequence of numbers collected at regular intervals over a period of time" then a time series database is a good choice (Namiot, 2015). For example, the open-source time series database InfluxDB[3] is suitable for SBC hardware and is

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

Table 4.2 Software features and examples of the three different types of data gathering construction.

	Type 1	Type 2	Type 3
Construction type	SBC with sensor(s)	SBC master node and group of SBM sensor nodes with sensors	Smartphone
Sensor software - code	Collect data from sensors - C/C++ or Python	Collect and send data - C/C++	Collect, store and send data - Android (service) program
Data gathering and preprocessing software	Python, suitable program library	Communication and processing, Python	The same as above
Data storage software	Filesystem	Filesystem or database	Filesystem, database or cloud storage
Data	Photos or similar large data chunks	Temperature, humidity or similar low data chunks	Photos or numerical sensor values

widely used in IoT solutions (Bader, Kopp and Falkenthal, 2017). Also, the relational database model is suitable for storing the collected data locally in a sensor device, for example a combination of Raspberry Pi with Linux OS, MariaDB[4] database, and a RESTful API. REST (REpresentational State Transfer) is an architectural style for distributed hypermedia systems (R. Fielding, 2000), (R. T. Fielding and Taylor, 2002). The RESTful API[5] (Application Programming Interface) is a web service which follows REST guidelines. With a RESTful API, the service allows remote control or management of a device over the network. There are other alternatives to the RESTful API technique, such as CoAP[6] and MQTT[7]. The study by Saari et al. (2020a) deals with the storage of sensor data more comprehensively.

Android smartphones have been used with this SW/HW framework. The Android OS software development kit (SDK) enables the wide use of smartphone capa-

[4]https://mariadb.org/
[5]https://restfulapi.net/
[6]https://www.rfc-editor.org/rfc/rfc7252.html
[7]https://datatracker.ietf.org/doc/html/rfc9431

bilities. For example, the SDK enables usage of a smartphone camera (Publication IV) and the smartphone's accelerometer sensor (Sillberg et al., 2018).

Also, Android OS facilitates data storage on smartphones via files and databases, enabled by the SDK. The SDK also enables users to utilize the data within the SW/HW framework, as well as transferring the data to cloud services. (Publication VI)

When considering data storage, cloud storage for data is a better choice than local storage in the master node because of security and availability. Cloud storage, such as the commercial Google Firebase or a self maintained Linux server, has more capabilities to store a larger amount of data than the local database in a Raspberry Pi. (Publication VI)

This section presented the developed SW/HW framework, as well as the prototype implementations. The SW/HW framework can be used as a guideline, with which it is possible to rapidly construct a data gathering prototype system for different purposes. The main limitation, but also advantage, is relying on off-the-shelf, open source, and community supported components. This is a limitation if producing a commercial product, but an advantage when rapidly prototyping a new data gathering system.

4.3 Use of the SW/HW framework with three types of systems

This section gives an overview of the developed prototype systems and lists the main findings made during the research. The prototypes are divided into Types 1 and 2, which are SBC and Linux OS based data gathering prototype systems, and Type 3 which is a smartphone based solution. (Publication VI)

In **Type 1** constructions, one or more sensors are connected directly to the SBC. The four different studies and main findings related to the SW/HW framework are presented briefly below. (Publication VI)

- Publication I presented the first implementation: a data collector service. In this prototype, a Beaglebone Black SBC gathered temperature, humidity, and brightness data. The data were stored in the SBC and server software offered the data to users. The SBCs used an Ethernet connection for data transfer. The main goal of the study was to test how well a cost-efficient off-the-shelf

SBC could collect, store, and provide data. This goal was reached successfully, and the designed system was tested and found to work as planned. The study proved that fully operating, data gathering prototypes can be developed with off-the-shelf devices and open source tools. (Publication I)

- The study by Saari, Baharudin and Hyrynsalmi (2017) focused on the usage of off-the-shelf devices when prototyping a sensor network solution in academic research. In addition, the study determined the advantages and limitations of prototyping when the Raspberry Pi was used. The study showed that the Raspberry Pi SBC is a widely used device in research implementations of different kinds. Some prototype system testing methods were found: software testing, software performance testing, and validation of data tests. The study by Saari, Baharudin and Hyrynsalmi (2017) clarified the operating environment for the SW/HW framework. (Publication VI)
- The third (Publication IV) and fourth (Grönman et al., 2019) studies were based on Raspberry Pi and camera combination prototypes. Both prototypes used 3G/4G communication. The data and photos were stored and processed in cloud storage. The SBC based prototypes collected and sent data for several weeks in the customer's environment, which raised the need for a remote control channel (for example the SSH service and terminal). (Publication IV)

A Type 1 construction typically consists of programs for data fetching, data checking, and data storing. An SBC environment has an OS and some programming language options for programming and implementing the software. The example use case, Program 4.3 uses Python programming language to read, modify, and store data. The function readData() reads two temperature values (the data) from an Arduino, which is connected with a USB cable[8] to a Raspberry Pi. The function writeData(values) modifies and stores data temporarily in an internal text file and sends the data to the Firebase Cloud Storage. This use case demonstrates data collection with a sensor, storing on the document database, and using a smartphone application, as described in the study by Saari et al. (2020a). The example is not a typical Type 1 construction because the sensors are used with an Arduino. The reason for selecting this kind of approach was educational - this example was used as a real-

[8]USB (Universal Serial Bus) is an industry standard for cables for the connection, communication and power supply between computers and peripherals.

world data gathering example in an Embedded Systems[9] course. Furthermore, the collected temperature data in the Firebase cloud service were used as an example in a course on Mobile Programming[10]. Both courses were held at Tampere University in 2019.

```
import time
from firebase import firebase
firebase = firebase.FirebaseApplication(
'https://kevat2019-b8e28.firebaseio.com/')
ser = serial.Serial('/dev/ttyACM0',9600, timeout=0)

def readData():
  data=ser.readline()
  temp1=''
  for value in data:
    temp1=temp1+chr(value)
  writeData(data)

def writeData(values):
  time_hhmmss = time.strftime('%H:%M:%S')
  date_mmddyyyy = time.strftime('%Y/%m/%d')
  data=date_mmddyyyy+';'+time_hhmmss+';'+arvot
  print(data)
  filename='Output.txt'
  with open(filename, "a") as text_file:
  print(data, file=text_file)

  #Data saving to Firebase
  line=data.split(';')
  try:
    result = firebase.post(
      'https://kevat2019-b8e28.firebaseio.com/Pikkuasevelitie'
      ,{'pvm':str(line[0])
      ,'Kello':str(line[1])
      ,'alalampo':str(line[2])
      ,'ylalampo':str(line[3])})
  except:
    print('Error')
```

[9]PLA-32311 Embedded Systems, 5 op
[10]PLA-32820 Mobile programming, 5 op

```
34      pass
```

Program 4.3 Python functions for data modifying and storing on Firebase Cloud Storage. First the Firebase Cloud Storage address is defined. The timestamp and collected measurements are combined into one line with a separator for local storing. In the last step "firebase.post" data are split and sent to the cloud. This prototype system is presented in a study by Saari et al. (2020a)

The **Type 2** construction was evaluated in three prototypes which had one SBC as a master node and several sensor nodes. These constructions use one-way communication from sensor nodes to the master node. This configuration was evaluated in several research cases:

- The first study, Publication II, concerned an Intel Galileo Gen 2 development board which functioned as a SBC master node. The sensor nodes were Arduino SBMs, which collected and sent data to the master node using wireless XBee[11] technology. The targets of the study were threefold: to test the multi node architecture model, to determine how well cost-efficient, off-the-shelf SBCs could be used to gather sensory data from several SBM sensor nodes, and how to deliver collected data to clients over the Internet. The study shows the usefulness of a Type 2 construction when several sensor nodes collect data from a small area (the size of area depends on the ability of the selected communication technology between the master node and sensor node). (Publication II)
- In the second use case, Rantanen and Saari (2020) presented a prototype system for monitoring indoor living or working conditions. The area of the sensor nodes was extended by using LoRa technology as a communication channel. In this prototype system, the Raspberry Pi master node with a LoRa expansion board received and stored data from several Sodaq sensor nodes[12]. The goal of the tests was to validate the prototype system construction. In addition, the study shows LoRa technology to be a good choice for sensor applications within concrete buildings (Rantanen and Saari, 2020).
- The third use case (Saari et al., 2020a) used a RuuviTag[13] consumer product

[11] Brand name for wireless connectivity modules, based on the IEEE 802.15.4-2003 standard designed for point-to-point and star communications

[12] https://support.sodaq.com/Boards/ExpLoRer/

[13] RuuviTag technical specifications, https://ruuvi.com/files/RuuviTag-tech-spec-2019-7.pdf

as sensor node. RuuviTags collect temperature, humidity, pressure, and motion data and use Bluetooth communication to send data to the Raspberry Pi master node. The master node has an InfluxDB[14] database for data storing, and Grafana[15] for data visualization. The advantage of ready-to-use consumer products is that the prototype system is quick to configure. The master node needs configuring, but the re-using of previously utilized software keeps the development effort low. These prototype systems showed that, even though the gathered data were small in quantity, the visualization required a lot of processing by the Raspberry Pi. The second issue raised was a restriction on the number of memory operations that could be performed using a memory card on a Raspberry Pi. Therefore the data had to be stored and processed on a cloud server. Regarding the SW/HW framework, this prototype system demonstrated the effectiveness of modular development and the interchangeable nature of both the sensor nodes and the master node. (Publication VI)

The software in Type 2 is divided into two types– sensor node and master node devices. The program 4.4 is an Arduino type solution for data gathering from sensors: First, the environment is configured in the setup loop. Second, two arrays are introduced in the loop: a "buffer" array variable for the message and a "tbuf" array for the sensor values. This construction consists of two sensors: SGP30[16] and BME680[17]. The data are fetched by means of functions. The last function "sendLoRa(buffer, strlen(buffer))" sends data to the master node using a LoRa wireless network.(Rantanen and Saari, 2020)

```
1  void setup()
2  {
3    setupLoRa();
4    setupBME680();
5    setupSGP30();
6  }
7  void loop()
8  {
9    char buffer[LORAWAN_MESSAGE_MAX];
10   if(!getSodaqBuiltInData(buffer)){
```

[14] https://www.influxdata.com/
[15] https://grafana.com/oss/grafana/
[16] https://learn.adafruit.com/adafruit-sgp30-gas-tvoc-eco2-mox-sensor
[17] https://learn.adafruit.com/adafruit-bme680-humidity-temperature-barometic-pressure-voc-gas

```
    return;
  }
  char tbuf[LORAWAN_MESSAGE_MAX];
  if(!getSGP30Data(tbuf)){
    return;
  }
  strcat(buffer, tbuf);
  if(!getBME680Data(tbuf)){
    return;
  }
  strcat(buffer, tbuf);
  sendLoRa((uint8_t*) buffer, strlen(buffer));
}
```

Program 4.4 Data gathering code for the sensor node. (Rantanen and Saari, 2020)

The master node software in the use case in Rantanen and Saari (2020) was implemented in a Raspberry Pi equipped with an additional LoRa shield device[18]. The program code was based on the GitHub project[19] described in Semtech S.A (2021) with several modifications. A pseudo code program 4.5 clarified the procedure of the master node functions. First, the message is received from a sensor node and checked. Then, if the message is approved it is divided into parts (timestamp, device ID, sensor value) and stored on the database.

```

loop{
  configurate LoRa device IDs
  Listen the LoRa communication
  If messages LoRa device ID is known
    saveMessage(message)
}
saveMessage(message){
  if message not ok
    return
    data = parseToSQLTable(message)
  send data to SQL
}
```

Program 4.5 Data processing and storing SQL code. (Rantanen and Saari, 2020)

[18]WiMOD Lite Gateway Data Sheet, https://wireless-solutions.de/downloadfile/lite-gateway-documents/

[19]https://github.com/Lora-net/packet_forwarder

Programs 4.4 and 4.5 describe the functioning of the data gathering system. This prototype system (Rantanen and Saari, 2020) showed the inconvenience of using ready-made code. The "Lora-net/packet_forwarder" GitHub project offers code examples, but the code implementation in one's own implementation is not always straightforward, due to missing code libraries or dependencies. Furthermore, the project's last commit (=update) was done in 2017 and there were only two maintainers for the project. However, the general conclusion remains that it is easier to modify existing code than to create entirely new code.

Type 3 uses smartphone related prototype systems because they are suitable for WSN sensor nodes. Smartphones come equipped with essential hardware components such as power source, communication capabilities, and sensor devices. Additionally, their operating system is well-suited for extensive utilization of the hardware. Our first attempt of using a smartphone as a sensor was presented in Sillberg et al. (2009). The research question of the study was "How to utilize mobile technology to supply disaster information to both mobile terminals and desktop computers?" (Sillberg et al., 2009).

Android smartphones were used in the two documented data gathering prototypes and their features are discussed from the perspective of the SW/HW framework. (Publication VI)

- The study by Sillberg et al. (2018) presented the initial prototype implementation for data collection, which relied on smartphone sensors, including GPS and an accelerometer, to identify changes in road surface conditions. An Android smartphone was the preferred choice for data acquisition as the study was conducted by a group of individuals who were driving on roads located in western Finland, and the majority of them possessed smartphones that met the requirements for prototyping purposes. The use of smartphones also focused on crowdsourced data collection. The developed software, ShockApp, was a combination of a user interface application and background service. The interface shows the state of the application and the background service tracks the location and gathers data from the sensors embedded in a smartphone. However, some issues arose concerning manufacturer-dependent features. Regarding the SW/HW framework, the prototype exhibited the capability of employing a smartphone as a sensor node for WSN. (Sillberg et al. (2018), Publication III)

- The second data gathering prototype was a solution for concept testing. The prototype system (software in a smartphone) consists of tracking a bus traveling on a route by collecting images, location data, and timestamps. The prototype is fully autonomous. The stops on the bus route were assigned to the prototype as GPS coordinate targets. The program code reads the GPS coordinates continuously and compares them to the assigned targets. When the coordinates match, a picture is taken. The working prototype solution for this problem is presented in Publication IV. For the SW/HW framework, this prototype again showed that a smartphone can be used as a WSN sensor node. (Publication IV)

The software differs significantly from that of Types 1 and 2. In Type 3, the Android smartphone environment is a sophisticated OS for mobile phones and it has several services and libraries available for software developers to use. On the other hand, the Android OS is limited due to security issues and the software has to be approved in order to use certain services with permissions. For example, the workflow of permission requests is clarified in programs 4.6 and 4.7. First, the necessary permission is declared - Camera in Program 4.6. After the permission configuration in 4.7, the program asks the user for permission to use the camera with a method (line 2) and collects the answer of the user with a method. When permission is approved, the software listens to the changes of service and, when a change of picture happens, the software should respond to the change if needed. After permission approval, a background function takePicture() is called whenever needed.

```
1 <uses-permission android:name="android.permission.CAMERA"/>
```

Program 4.6 A Permission declaration in manifest.xml configuration file

```
1 private boolean checkCameraHardware(Context context) {...}
2 private void requestCameraPermission() {
3     if (shouldShowRequestPermissionRationale(Manifest.permission.
      CAMERA)){
4         ActivityCompat.requestPermissions(
5             this, new String[]{Manifest.permission.CAMERA},
6             REQUEST_CAMERA_PERMISSION);
7     } else {
8         requestPermissions(new String[]{Manifest.permission.CAMERA
      },
9             REQUEST_CAMERA_PERMISSION);
```

```
    }
}
 @Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0]
                != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.
    CAMERA},
                REQUEST_CAMERA_PERMISSION);
        }
    } else {
        super.onRequestPermissionsResult(requestCode,
            permissions, grantResults);
    }
}

protected void takePicture() {
    CameraManager manager = (CameraManager)
        getSystemService(Context.CAMERA_SERVICE);
        ...
    ImageReader.OnImageAvailableListener readerListener
        = new ImageReader.OnImageAvailableListener() {
             @Override
            public void onImageAvailable(ImageReader reader) {
                ...
            }
    }
}
```

Program 4.7 Java file for Android environment and permission request.

Program codes 4.6 and 4.7 give a certain example of coding in an Android OS environment. The environment contains dozens of sensors[20] (depending on the device) and it has all the necessary features for the working sensor device (communication, battery, and programmable environment). Therefore, if values can be measured with a smartphone, it should be used.

[20]https://developer.android.com/reference/android/hardware/Sensor

4.4 Discussion and summary

This chapter outlines the impact and significance of the framework, and aimed to resolve the research question: **RQ2:** How can IoT data gathering be generalized into a framework of required software and hardware components? To answer the question, several data gathering prototypes were made.

The research and the prototype system development projects presented above show that it is possible to categorize prototype systems into three different types of construction depending on the use case. Type 1 constructions are suitable for large amounts of data from a few sensor type systems, for example camera applications. In addition, Type 1 prototype systems are freely modifiable unlike Type 3 systems. The Type 2 construction collects simple data from several points with separate sensor nodes, for example living conditions from a residential building. In Type 3 constructions, a smartphone is used as a data gathering sensor device, and this is a suitable construction if the smartphone's sensors are appropriate for the selected use case.

In the realm of developing IoT data gathering systems, studies have highlighted several aspects regarding the construction of a SW/HW framework. The utilization of off-the-shelf devices offers significant advantages, particularly in terms of lightweight development and cost efficiency. These devices, including readily available hardware like Android smartphones and Raspberry Pi Single Board Computers (SBCs), provide a versatile and cost-effective solution for rapid prototyping and development. (Publication V)

The operating systems of the selected devices support the usage of sensors, further enhancing their applicability in IoT environments. Off-the-shelf devices enable a quick start to development, eliminating the need for extensive custom hardware design, thereby reducing the initial time and investment required. This approach is especially advantageous in the early stages of a project, where the focus is on validating ideas and concepts through prototyping. (Publication IV

The idea of the framework is not new and there have been a few closely related studies (Kreiner et al., 2001), (Srivastava and Brodersen, 1991), (Mäntyniemi, Pikkarainen and Taulavuori, 2004). Nevertheless, the main idea was to give a new perspective on developing a framework for data gathering using off-the-shelf components, both software and hardware. The related research provided the building blocks, e.g., Xu, He and Li (2014) and Vakaloudis and O'Leary (2019) introduced

IoT layers: sensing, networking, service, and interface, which fit our framework architecture. In addition, Chapter 3 and the design factors (Akyildiz, Sankarasubramaniam and Cayirci, 2002) provided a basis for developing the SW/HW framework.

The SW/HW framework development raises several new research topics. As the first topic, all the prototypes discussed in this context feature sensor software, which is responsible for reading the sensor's output. For instance, in the case of a temperature sensor, the software converts the byte value into an integer using a suitable formula, and either sends or saves the data at an appropriate location. Sensor software is mentioned several times in this thesis, but its algorithm has not been discussed in detail. This low-level algorithm is programmed using C/C++, Python, Java, or a similar programming language. It should be noted that the initial data processing could be performed at this level, e.g., averaging the accelerometer sensor values within one second. How should low-level software be programmed in order to improve performance without data loss? (Publication VI)

For the second topic, a large amount of sensor data were collected by the prototypes. Data processing and data mining are important aspects, which this study leaves for future research. Data visualization has been touched on briefly in the studies by Sillberg et al. (2018) and Saari et al. (2020a). In addition, sensor data will become more usable if merged with other publicly available data such as weather data or map data (Sillberg et al., 2018), (Soini et al., 2019).

The third topic would be performance issues, which have not been extensively addressed in this research. In the Type 2 data gathering construction, the amount of sensor nodes is limited, but no exact limit can be set because it depends on the range, communication channel, transferred data, and so on. Performance problems may also be affected by data processing such as in the case of photos or especially motion detection. This raised the point of what should be processed in SBC devices and what should be processed in the cloud service? (Publication VI)

The quality criteria for components is not defined in the SW/HW framework. This raises the question of how to select good components and devices for prototypes. This question is left to the framework user.

The vulnerability issues of the data are worth considering. What happens if data are not available and second, if the data are critical? What would the consequences be if the data were manipulated? The SW/HW framework ignores the matter, but these are significant issues. Furthermore, security issues for IoT devices are important to

note. The study by Babar et al. (2011) discusses security vulnerabilities and attacks on IoT systems. (Publication VI)

The SW/HW framework was presented in this chapter as the second artifact of the research. It was proved in this chapter that the architecture models presented previously in Chapter 3 can be extended to the SW/HW framework, which more precisely determines the guidelines for constructing an IoT data gathering prototype system. With the SW/HW framework guidelines, it is possible to rapidly construct a data gathering prototype system for different purposes. Furthermore, this SW/HW framework together with Chapter 3 "Architecture model for sensor node" provides a base for developing a model for the prototyping process, which is presented in the next chapter.

5 MODELING THE PROTOTYPE DEVELOPMENT PROCESS

The purpose of this chapter is to discuss the development of the prototype development process model, the Descriptive Model for the Prototyping Process (DMPP), described in Publications VI-VIII.

During the research phase of the thesis, several data gathering prototypes were developed using the same development process. The focus of this chapter is the description and assessment of the prototype development process itself. This chapter answers the research question presented in the introduction: **RQ3:** What kind of process model can be developed from prototype development practices that have been applied in research projects between university and enterprises?

The subject is approached by introducing a modeling procedure with example prototype pilot cases. An eight-step process modeling approach was utilized to recognize instances of activity, artifact, resource, and role. With these instances, the artifact DMPP was developed.

The main result of this modeling – the developed prototype development process model – can be used as a guideline when building prototype systems with a client. An evaluation of the DMPP is presented in Publication VIII and Section 5.3.

5.1 Developing the process model

During the research it was noticed that the same procedure was followed when conducting the process. This observation led to a further development idea: How to model the development process? The prototype development projects and their results demonstrated the ability to execute projects quite effectively. Also, both the customer and the project group were satisfied with the results, i.e., prototypes, documents, and data.

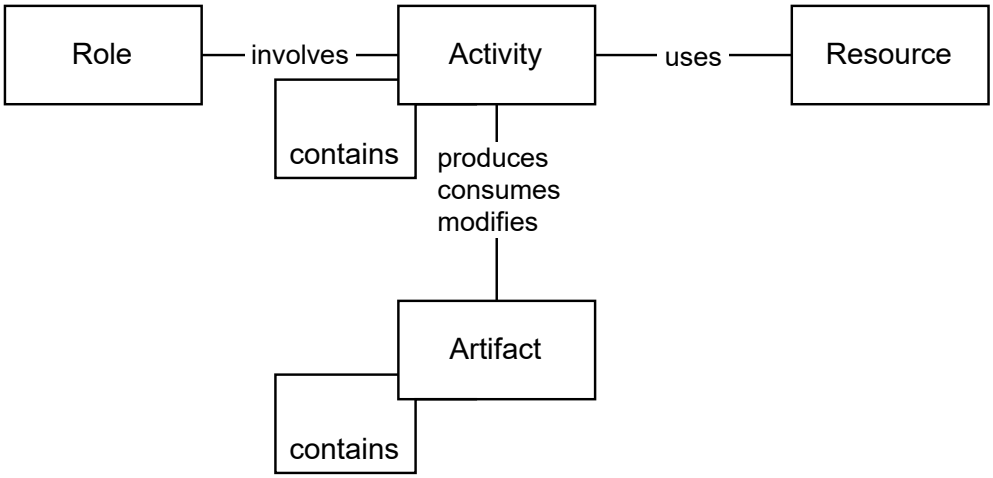

Figure 5.1 Fundamentals of the prototype development process. Adapted from Publication VII.

In this section, the aim is to present the creation process of the DMPP. The Descriptive Process Model (DPM) approach proposed by Becker, Hamann and Verlage (1997) was followed. This approach consists of eight steps grouped into two phases:

Setup phase

1. Objectives and Scope
2. Define Schema
3. Select Language
4. Select and Tailor Tools

Execution phase

5. Elicitation
6. Create Model
7. Check Model
8. Check Process

The DPM approach was applied in the following way: To collect data for the models, the developers involved in the processes were interviewed. The schematic diagram shown in Figure 5.1 guided the data collection, the results of which were shown on stickers on the wall during the work (Figure 5.2). In the resulting model

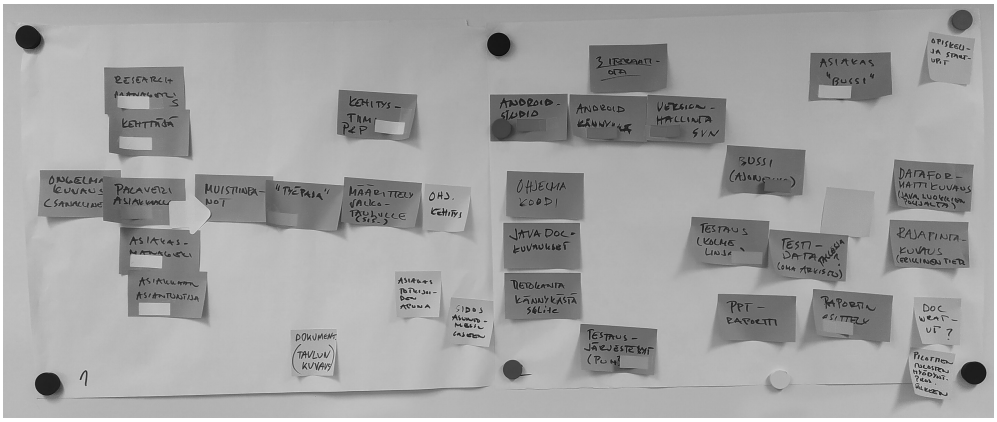

Figure 5.2 The modeling of the bus case development process with whiteboard and notes. The bus case prototype system is introduced in Publication IV.

(Figures 5.3 and 5.4), the activities are represented as rectangles with rounded corners. The roles are represented by stick figures and resources are represented by different icons. Parallelograms, cylinders, and document symbols represent artifacts. Continuous arrows represent the associations between activities and artifacts, while the links between activities and roles and resources are dashed. Graphical representations of the models were produced using free online diagram software: draw.io[1]. (DPM Steps 2-4). (Publication VII)

The objectives and scope of the modeling were presented in the introduction of this chapter (DPM Step 1). The following section provides an example of data elicitation (DPM Step 5) and the modeling results (DPM Step 6). After that, the possibilities for improving the modeled processes are discussed (DPM Steps 7-8). (Publication VII)

The DPM approach handles process elicitation in Step 5 and creates the process model in Step 6. This section highlights the process knowledge by introducing one of the prototype development processes – the bus case – as an example. The bus case prototype system is described in Publication IV. The development process practices can be highlighted using a whiteboard and Post-It notes (Figure 5.2) (Raninen et al., 2013)]. Publication VII presents three other prototype development processes, which were used as a knowledge base when developing the DMPP.

The bus case was established to address customer complaints received by a bus

[1]https://app.diagrams.net/

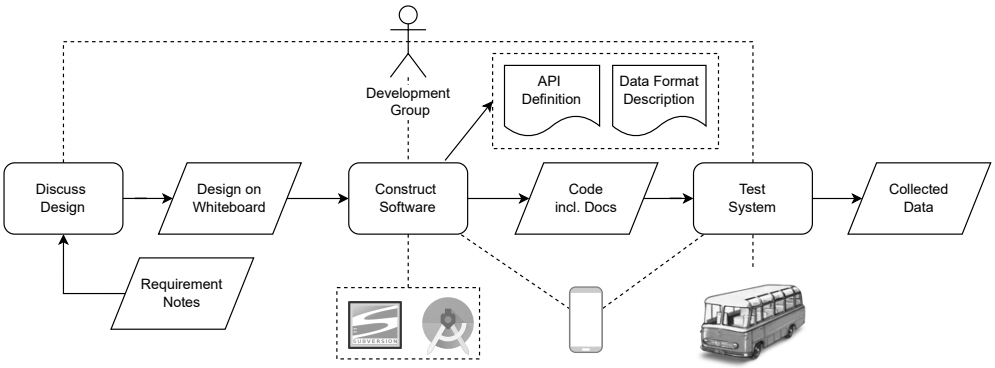

Figure 5.3 Bus case development process. The project group developed a working prototype introduced in Publication IV. Adapted from Publication VII.

company. Customers had complained that buses were not stopping to pick them up or not coming at all. To collect photos with timestamps at specified bus stops, a prototype was developed and implemented on mobile phones. The main idea of the prototype was to collect photos of the bus stops as the bus approached. The university's project group developed a working prototype, which was then tested in the buses. (Publication IV)

The bus case prototype system development process is illustrated in Figure 5.3. The development group in this process consisted of the university's research group. The development process began with a design discussion, which was the first activity (rounded corner rectangle) that produced the first artifact (parallelogram): the whiteboard sketch. The artifacts or results were subsequently utilized in the software construction phase activity. (Publication VII)

The activity steps "Construct Software" and "Test System" were later combined into the "Develop Software" activity (Figure 5.4). The resources used during the development process, including the programming language, test device with GPS, camera, and network, were involved in the activity step. The activity step "Test System" included the testing environments: the laboratory and the bus itself. The artifact step "Code including Docs" consisted of the prototype device and software code with documentation. It also included the API definition and Data Format Description. After that the testing activity started, which produced the collected data artifact. The coding and testing activities could be iterated several times. (Publication VII)

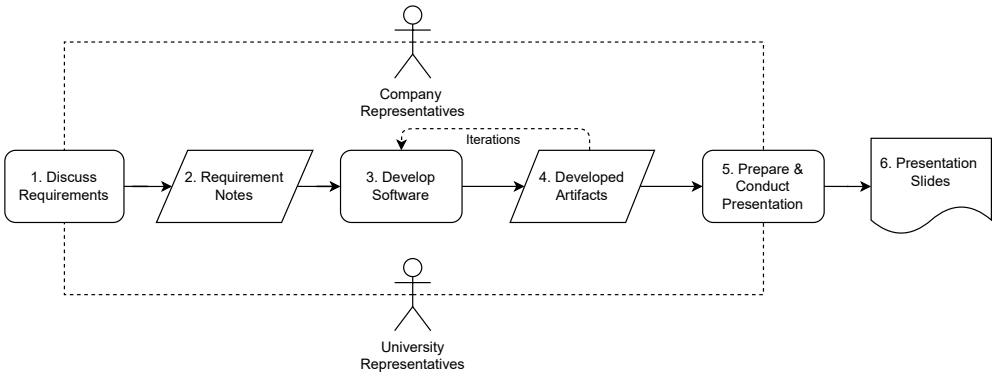

Figure 5.4 Process model for prototype development. The rounded corner rectangles represent activities and the parallelograms represent artifacts. Adapted from Publication VII.

The presented bus case development process produced a working prototype. In this case, the prototype was then presented to the customer, i.e, the bus company. (Publication VII)

5.2 The DMPP

The Descriptive Model for the Prototyping Process (DMPP) is a result of DPM Step 6. The process model describes the prototype development practices that were applied in research projects between university and enterprises. Figure 5.4 presents the developed DMPP and gives an overview of the steps included in the process. The process model developed seeks to present the methods and practices used in prototype development projects in general terms.

The DMPP is not limited to a specific type of artifact, so it can be extended to include hardware. Therefore, in this thesis, the term "process" generally refers to all types of artifacts, whether software or hardware, produced in the prototype development process. Although the DMPP primarily models the prototyping process within a research context, it is not restricted to this context. (Publication VI)

The fundamental concepts related to processes are role, activity, resource, and artifact. For example, in a software development activity, a developer (role) uses a programming tool (resource) to create software (artifact), which is later employed in the prototype system. (Publication VII)

The DMPP is based on the six main steps (Figure 5.4) presented below. The use

of previously presented architectural models and the SW/HW framework is also described in steps (Publication VII):

1. The **Discuss Requirements** step is initiated with the collaborative requirements definition discussion between the developer group and the client. During this discussion, the client specifies the type of data they require, while the developer group establishes the hardware and system architecture, as well as how the software will collect the data. The choice of hardware mainly influences the software environment and tools that will be utilized.
2. The first artifacts, **Requirement Notes**, are the outcome of the discussion: the first architecture model of the component interconnections, which includes the prototype system requirements within the discussion notes. The thesis focuses on the idea of selecting an architecture model between single node or multi node, but naturally more architecture models are possible. (Publication VI)
3. The **Develop Software** step involves the research group, including the project manager, in the development of the software/hardware prototype. The client's representatives are involved in the development process in the role of instructor. In this step, the selected architecture model and the SW/HW framework are used as the guidelines for selecting the components for the developed prototype. (Publication VI)
4. The **Development Artifacts** step introduces the working prototype artifact. It contains the developed software and hardware components. The interconnections of the components are tested, and the prototype system is functionally tested. The gathered data are also inspected and, if possible, compared with the expected results. Iterations are possible especially in situations where the prototype functionality has been tested in a laboratory environment before the field tests. (Publication VI)
5. The **Prepare & Conduct Presentation** step includes preparing the outcome of the development process. Possible results include a prototype system and documentation, collected data, and project documentation. Also, the SW/HW framework can be complemented if necessary. (Publication VI)
6. The **Presentation** step is to publish the results, for example, the prototype system, collected data, and analysis of the project. The audience for the publi-

cation is at least the client, and in most cases the results are freely available for further development. (Publication VI)

Prototyping is a powerful way to figure out the possibility of implementation. Other benefits of prototyping include cost estimation and workload estimation. The process model shown in Figure 5.4 is a simplified presentation of the prototype development process and therefore it does not mention common procedures such as iterations, testing, and customer testing (Publication VI).

When analyzing the process model and the selection processes (DPM Steps 7 and 8), the lack of iteration notation arose. Iteration is used in most prototype development projects when configuring, constructing, or testing a prototype. The use of iterations is an efficient way to test and develop an idea. The first iteration starts with a basic prototype, which consists of simple and basic components. For instance, the hardware could be chosen solely for the purpose of testing the idea. If the initial prototype works, subsequent iterations improve upon it by replacing the basic components with more suitable ones, including hardware and software components. Iterations can help identify flaws and limitations, and thus refine the prototype until it meets the requirements. (Publication VI)

5.3 Evaluation of the DMPP

Publication VIII focuses on the KIEMI research project and its use of the DMPP to facilitate collaboration between university and enterprises. The project included 23 pilot case projects where prototypes were developed in collaboration with enterprises to address real-world issues. The term "pilot case project" is used here to distinguish individual prototype projects from the KIEMI project. This section reviews Publication VIII which evaluates the suitability of the DMPP for usage in a research project.

Clarifying the benefits of the different stages of the DMPP and their applicability to collaboration pilot case projects, the following points are discussed:

Discuss Requirements: Most prototype development pilot case projects involve an external partner in discussing objectives, with varying levels of collaboration. In low-level collaboration, the partner provides the premises for measurements without making any special requests. The output is typically a report that may or may not lead to further actions.

In mid-level or high-level collaboration, the partner takes a more active role in discussions and directs the starting point towards a specific issue they want to research. High-level collaboration often involves expanding the original task assignment and bringing in additional partners or stakeholders.

The DMPP is well-suited for this type of activity because the non-commercial leader, i.e., the university research team, is focused on research goals rather than financial goals. Additionally, any additional research or technical goals set by partners are shown to be applicable to the model's operation within the iteration rounds. In these cases, the university research team led the pilot case project and collaborated with necessary partners. (Publication VIII)

The main purpose of **Requirement Notes** is to guide the pilot case project in the selected direction, making them an important part of the documentation. The DMPP demonstrates the advantage of "light documentation" for getting started quickly, using previously defined architecture models and device configurations to speed up operation. This approach also involves reusing technological choices and definitions from earlier pilot case projects, based on the idea that "Some Things Are Better Done than Described" by Hunt and Thomas, 2000.

Light documentation and process modeling are particularly suitable for university and other research institution environments focused on prototyping, rather than commercial product development. However, this approach may require more work if technology transfer to a partner begins with a prototype. Internal requirements are also mentioned in several cases, such as when the research group wants to change or update a specific feature. (Publication VIII)

The **Develop Software** phase uses the artifacts of previous requirements as a loose guideline. For example, user interface software by Nurminen et al., 2021 and backend software by Nurminen, Saari and Rantanen, 2021 developed in early stage pilot case projects were used in several subsequent pilot cases. The DMPP allows for changes to requirements if they are deemed beneficial. These changes are not typically discussed with partners unless their input is required. While the DMPP does not specify requirements for software or hardware components, it was observed that using off-the-shelf components accelerated prototype development. Additionally, this approach offers the advantage of flexibility in adapting prototype solutions to conform to the requirements of selected components. (Publication VIII)

The main goal of the DMPP phase is to develop fully working prototype sys-

tems, which are considered as **Development Artifacts**. In the KIEMI project, this phase typically involved installing the prototype at a target location provided by the partner to collect data. The majority of the prototypes were working SW/HW prototypes, although some were only SW prototypes used for analyzing and visualizing the partner's collected data. The DMPP aims to produce a functional prototype, and as such, only the primary functions of the prototype are utilized, with documentation or testing done only partially. While this approach speeds up development, it could potentially slow down technological transfer later on. (Publication VIII)

The phase of **Prepare & Conduct Presentation** is intended for reporting the results of the pilot case project. In longer pilot case projects, it was observed that the reuse of skeleton reports accelerated this phase of the pilot case project. The automation of this process significantly sped up the reporting phase. This highlights the fact that when using the DMPP model, reporting will typically consist of the same components. (Publication VIII)

The final phase of the DMPP involves **Presenting** and publishing the pilot case project results. In successful pilot case projects, partners often express interest in further developing the prototype, and technology transfer continues from this point. One significant advantage of the DMPP model is that it ultimately aims to publish scientific and other public material from the pilot case projects. (Publication VIII)

Overall analysis: The KIEMI project demonstrated the usability of the DMPP and its suitability for pilot case projects. The project utilized two approaches: software development style and collaboration style. The software development style emerged during the prototype system development with the goal of implementing new systems to collect, process, and present environmental data. The collaboration style was a result of cooperation with various partners during the prototype development process. The DMPP is able to connect both styles. (Publication VIII)

The project successfully demonstrated collaboration between a university and enterprise in the context of prototype development. In most cases, the DMPP process was in the background and invisible to the partners, but it provided support for collaboration throughout all six phases. The DMPP can support technology transfer, especially in phases 1, 3, 4, and 5, where cooperation with the partner is necessary. (Publication VIII)

To improve collaboration, it is beneficial to add a step where the company provides a suitability assessment of the prototype's general level and associated return

on investment (ROI). With the feedback received, the research team can accumulate expertise in designing the next prototype and produce a result that is more interesting to the company. The ability to rapidly produce prototypes valued by companies is a significant strength and advantage for a university that organizes projects. (Publication VIII)

5.4 Discussion and summary

This chapter's related studies in Chapter 2 started with a discussion about software development processes: Waterfall, Agile, and rapid prototyping. Of these, rapid prototyping was the closest to the process model. However, rapid prototyping did not include everything we needed as a definition, e.g., customer contact or presentation of final outputs, so we developed our own. The process of model development is strongly related to prescriptive and descriptive process models. A prescriptive model describes how the process should be performed, whereas a descriptive model describes how a process is performed in a particular environment. The outcome of this chapter is the process model that was developed in a descriptive way. Previous prototype development projects were used to develop the model.

Another related model – the DPM process model – was used as a guideline for developing software. This chapter showed the ability of the selected models and methods to build up the DMPP. In addition, using the three major parts of developing embedded systems – model-design-analysis (Lee and Seshia, 2017) was a good approach: first, select the data to collect, next build a prototype to collect the data, and finally, analysis, to determine whether the outcome was successful.

This chapter introduced the DMPP – a model for the prototype development process by modeling a procedure with example prototype cases. The research method used and presented here was an eight-step process modeling system. The basic concepts relating to the prototype development process included four factors: activity, artifact, resource, and role. The outcomes of the modeling procedure were conveyed through textual and graphical representations, and the process knowledge obtained during the model creation was described. Furthermore, certain shortcomings in the existing practices were discovered. Based on the findings, both the model and the prototype development process and practices could be further improved in the future. (Publication VII)

The DMPP was developed to guide the prototype development process. The process model in Fig. 5.4 is a representation of the prototype development process. It gives the abstract instructions for the operation with defined steps to implement data gathering IoT prototypes from start to finish. If all steps are performed, the level of the outcome is predictable. The model is sufficient for developing prototypes, but it also makes it possible to add more activities if needed. For example, procedures such as iterations, testing, and customer testing could be included in the process.

The model presents the basic concepts: customer and developer team **Roles**; the **Activity** phases of the project; **Resources**, e.g., the software and hardware to use; and **Artifacts** such as notes, software, hardware, prototype, published reports, and presentations. More specifically, it provides the necessary steps for handling requirements and specifications, including the implementation phase and the publishing of the developed prototype. When the model steps are followed, this produces the artifacts: notes, prototypes, and collected data, and finally the published results of the project. If the developer team, or at least the project manager, knows these concepts the project can be expected to be successful.

The modeling work environment presented in this chapter was focused on a university environment and similar research organizations. This raised the question, which could even be the research question of the thesis: How to generalize the use of the process model and the development process? Furthermore, the prototypes were developed with quite a small group – even when customers were included the number of participants was below twenty. What if the number of participants were fifty for example, would the model still be applicable? These questions must be taken into account in the selection of future research topics.

When considering the process model research and validating the results presented in this chapter, the question of timeline arises, and more specifically the question: Is the process model the desired output of the thesis or is it a side result of the overall thesis research project? The article "Threats to validity of Research Design" points out the "reactive or interaction effect of testing" thread, where pretest arrangements jeopardize the test input and results (Shadish, W., Cook, T., Campbell, T. (2002) as cited in Chong-ho (2021)). To prevent this effect, previously executed case studies were selected when developing the DMPP.

In summary, the DMPP provides a concrete and systematic example of how collaboration between university and enterprise can be executed in practice. Thus,

based on the observed results, the model can be considered successful and fit for the needs of the development cases in question.

6 CONCLUSION

This thesis began with an introduction of the DS method and the six phases it contained (Figure 1.4) that were used during the research. At first, the problem was identified and the research questions set, and the scope was defined. The design and development phase was described in Publications I - VII. Chapters 3-5 presented the artifacts: the architecture model, SW/HW Framework, and DMPP. The evaluation of the work was done when the publications went through the review processes. This is part of the last phase of the research process: communication.

This final chapter summarizes the thesis research work and presents the final conclusions of the outcome. This chapter revisits the research theorem, discusses its selection, and how it was approached. The research theorem was developed further into three research questions. These questions and how the answers were found are also revisited. Furthermore, the limitations of the presented models, framework, and process are discussed. Finally, future directions for research are introduced.

6.1 Revisiting the research questions

The research theorem and the main question in the thesis was: How to construct a system architecture model of wireless sensor network nodes and process models for the prototyping process to efficiently develop data gathering for IoT applications? The answer to the question is divided in this thesis into three chapters: 3, 4, and 5. The thesis chapters are constructed by dividing the main question into three subquestions.

Chapter 3 focused on answering the research question: **RQ1:** What kind of sensor model architecture can be developed for data gathering in a wireless sensor network? The research started by exploring the subject and by constructing data gathering prototypes in the period from 2014 to 2018. The results were first published in Publications I – III.

The architecture model contains software and hardware components, and their interconnections. Two different architecture models: **multi node** and **single node** have been presented. The architecture models present the basic approach when constructing the first implementations of data gathering prototype systems using off-the-shelf components, hardware or software. The use of these fundamentally simple models enabled the creation of highly practical and interoperable sensor applications to gather data on environment conditions. Furthermore, the models provided a strong background to start the development of the SW/HW framework.

The focus of Chapter 4 was on answering the question **RQ2:** How can IoT data gathering be generalized into a framework of required software and hardware components? The research started by collecting and summarizing the features of the data gathering prototypes. Three publications, IV – VI, built up a general framework and, in addition, Chapter 4 summarized the different aspects. Chapter 4 introduced the software / hardware framework for IoT data gathering. The SW/HW framework generalizes data gathering prototype development into a group of required components and, more precisely, the framework defines guidelines for constructing prototype systems to collect data for different purposes. The framework presents three concrete examples (Types 1-3) with software and hardware components to help developers to get started. The lessons learned from the SW/HW framework are suitable when starting to develop a data gathering prototype. With these instructions, the development process can be guided toward selecting the components for constructing a prototype system.

Chapter 5 answered the last question **RQ3:** What kind of process model can be developed from prototype development practices that have been applied in research projects between university and enterprises? The method for this research was an eight-step process modeling approach, which was selected from the software process modeling area. The selected research method gives reasonable solutions to the research problem.

The last research question **RQ3** was examined in publications VI-VIII. Publication VII introduced the prototype development process in a university environment. The prototypes were made in collaboration with companies, which offered real-world use cases. The prototype development process was introduced using a modeling procedure with four example prototype cases. Publication VI combined the development process from Publication VII with the SW/HW framework and

showed that the framework and process model could be used together. Publication VIII showed that the model is practically usable in collaborative environments, particularly in enterprise-university research projects and can support such developments in similar environments.

At the beginning of the thesis, Figure 1.3 illustrated how the research questions and contributions were connected to each other. The first subject to discuss when a project group follows the DMPP is the requirements. At this point, decisions on the collected data and further on the architecture model of the data gathering device are needed. The architecture model guides the selection of SW/HW framework data gathering device type. The project group can continue to develop the system after these fundamental decisions have been taken. The rest of the DMPP should proceed flawlessly with the selected technologies. If not, the DMPP gives the possibility to return and perform a second iteration round, as was done in Publication IV, where the original technology (smartphone) was switched to another (Raspberry Pi) due to a reliability problem.

6.2 Contributions and summary

The thesis scope and contributions were described in Chapter 1.3. The methods and approaches proposed in this thesis were implemented and experimentally verified by prototyping different kinds of data gathering systems. The results of the research have been documented and presented to the scientific community. The contributions of this thesis are as follows:

- Single node and multi node architecture for guiding the design of data gathering IoT sensor devices.
- The SW/HW framework for guidance when selecting suitable components for building data gathering IoT sensor devices.
- The DMPP, which allows the guidance and control of IoT prototype development projects in university - enterprise collaboration.

Chapter 3 focused on the architecture model for the data gathering sensor node and presented a concise overview of the subject. The main contributions of the chapter are the **multi node** and **single node** architecture models. The multi node

architecture model is designed to collect simple environmental data with an unlimited number of sensor nodes. These sensor nodes are controlled by the master node, which controls and manages the sensor nodes. The single node architecture model was developed to represent more complex data acquisition systems.

The main components of both models were described: sensor node, master node, communication, and the Internet as a communication channel for user applications. Furthermore, the purpose of the models and possible applications as a data collection tool were presented. The possibilities for the architecture models are much more than the two presented instances. Nevertheless, with these two architecture models the thesis subject remained sufficiently narrow; additionally, these two provide a strong background to Chapter 4.

In Chapter 4 the framework for IoT prototype development was presented and defined – several prototypes were built and the framework is the generalization of the software and hardware components used. The main contribution of the framework is the generalization of the necessary components. The software components for collection, preprocessing, and storage were presented. The hardware components for different purposes were also presented. For a developer, the framework gives a starting point for building a data gathering prototype system.

Chapter 5 introduced the DMPP to guide the prototyping process. The main contribution is a concrete process model of prototype development for university – enterprise collaboration in practice. The development process was examined and analyzed by introducing the roles, activities, resources, and artifacts involved. Further, it described how the prototypes were developed independently or with an external partner.

6.3 Limitations and future work

Chapter 3 focused on the architecture model and Chapter 4 focused on the SW/HW framework. The hardware to test both of these was based on a selected set of off-the-shelf devices, with limited variation. Furthermore, no quality requirements (e.g., performance) were set for the hardware components. Therefore, in future the variation should be expanded, and the quality requirements would have to be defined to find more suitable devices.

Chapter 5 introduced the DMPP for guiding prototype development in collabo-

ration between university and enterprises. Also, an evaluation of the process in an academic research project was presented. However, it did not describe how to generalize the process to the real world – outside a university or other research institution environment.

Thesis was closely linked to prototypes and projects, though its primary aim was not to delve into the commercialization of these prototypes. A key goal in these projects was to disseminate developed technology to the general public and for corporate use. We frequently discussed within project teams how small and medium-sized enterprises, and startups should manage the prototype for its continued development and support. Despite these discussions, we did not find an easy, clear-cut solution. Nevertheless, several prototypes were eventually deployed in production. An interesting aspect of these production-deployed prototypes was the significant involvement of the client or company in identifying issues and participating actively in the prototype's development(e.g., (Rantanen et al., 2021)).

As future work, ways should be found to extend and utilize the architecture node model, the SW/HW framework, and the development model of the prototyping process in new areas of the IoT environment.

The architecture model described in Chapter 3, and the SW/HW framework in Chapter 4, are based on the idea of building sensor nodes, but what if reasonable (and open-source) off-the-shelf devices were used? This kind of approach has already been tested in an ongoing research project where RuuviTag sensor nodes were used for data collecting and the research focused on the construction of an architectural system with data storage, visualization, and analysis (Nurminen et al., 2021), (Nurminen, Saari and Rantanen, 2021). One of the research conclusions from those studies was that there are several ready-to-use components available for an IoT system developer. In addition, these two models were designed for general purposes, and specific industrial cases were avoided. In the future, one possible variation of the models may be for example an evaluation of industrial data collection.

The main focus of the thesis was the testing of the ideas of models, architectures, and processes. In the future, there will not be any limitations to support the theory of the thesis and the use cases can be expanded to new fields. One such example is the study by Rantanen et al. (2021) where the basic configuration described in Chapter 4 represented a Type 1 construction - sensors connected directly to an SBC. The specialties in this case were the hardware and software: the hardware was an

NB-IoT capable SBC and the software was based on libraries from the hardware user community. In these cases, the goal was low power and long-term usage. The study shows that the basics of this thesis are suitable for a variety of use cases.

All of the data gathering prototype development processes have been based on light documentation. This could have been due to the laziness of the developer group but, as stated in Hunt and Thomas (2000) (page 218), "Some Things Are Better Done than Described": it was a working method used in data gathering prototype development. The light documentation and the process modeling environment focused on university and other research institution environments where the focus was on prototyping rather than the development of commercial products. One further study topic could be: How to generalize the use of the process model and the development process so it could be used in the business world?

REFERENCES

Akyildiz, I., Sankarasubramaniam, Y. and Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine* 40.8, 102–114. DOI: 10.1109/MCOM.2002.1024422.

Android (operating system) (2021). Accessed February 26, 2021. URL: https://en.wikipedia.org/wiki/Android_(operating_system).

Arduino (2020). *Arduino - Home*. Accessed December 21, 2020. URL: https://www.arduino.cc/.

Ashton, K. (2009). That 'Internet of Things' Thing. *RFID Journal*. Retrieved 28th of February, 2023. URL: https://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf.

Atlam, H. F. and Wills, G. B. (2020). IoT Security, Privacy, Safety and Ethics. Springer International Publishing, 123–149. DOI: 10.1007/978-3-030-18732-3_8. URL: http://link.springer.com/10.1007/978-3-030-18732-3_8.

Atzori, L., Iera, A. and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks* 54.15, 2787–2805. DOI: 10.1016/j.comnet.2010.05.010.

Babar, S., Stango, A., Prasad, N., Sen, J. and Prasad, R. (2011). Proposed embedded security framework for Internet of Things (IoT). *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*. IEEE, 1–5. ISBN: 978-1-4577-0786-5. DOI: 10.1109/WIRELESSVITAE.2011.5940923. URL: http://ieeexplore.ieee.org/document/5940923/.

Babich, N. (2019). *What is Rapid Prototyping?* Accessed November 16, 2020. URL: https://xd.adobe.com/ideas/process/prototyping/rapid-prototyping-efficient-way-communicate-ideas/.

Bader, A., Kopp, O. and Falkenthal, M. (2017). Survey and comparison of open source time series databases. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)* 266, 249–268. ISSN: 16175468.

Baharudin, A. M. bin, Saari, M., Sillberg, P., Rantanen, P., Soini, J., Jaakkola, H. and Yan, W. (2018). Portable Fog Gateways for Resilient Sensors Data Aggregation in Internet-less Environment. *Engineering Journal* 22.3, 221–232. ISSN: 01258281. DOI: 10.4186/ej.2018.22.3.221.

Baharudin, A. M. bin, Saari, M., Sillberg, P., Rantanen, P., Soini, J. and Kuroda, T. (2016). Low-energy algorithm for self-controlled Wireless Sensor Nodes. *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, 42–46. ISBN: 978-1-5090-3837-4. DOI: 10.1109/WINCOM.2016.7777188.

Barnett, R. H., O'Cull, L. D. and Cox, S. A. (2003). *Embedded C Programming And the Atmel AVR*. Thomson, 495.

Barr, M. (2020). *Embedded Systems Glossary.* Accessed December 14, 2020. URL: https://barrgroup.com/Embedded-Systems/Glossary.

Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M., Couach, O. and Parlange, M. (2008). SensorScope: Out-of-the-Box Environmental Monitoring. *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IEEE, 332–343. ISBN: 978-0-7695-3157-1. DOI: 10.1109/IPSN.2008.28. URL: http://ieeexplore.ieee.org/document/4505485/.

Becker-Kornstaedt, U. and Webby, R. (1999). A comprehensive schema Integrating Software Proces Modeling and Software Measurement. *IESE-Report No. 047.99/E*.

Becker, U., Hamann, D. and Verlage, M. (1997). Descriptive Modeling of Software Processes. *IESE-Report No. 047.97/E*.

Brooks, F. P. (1995). *The Mythical Man-Month (Anniversary Ed.)* USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201835959.

Brown, A. and Short, K. (1997). On components and objects: the foundations of component-based development. *Proceedings Fifth International Symposium on Assessment of Software Tools and Technologies*. IEEE Comput. Soc. Press, 112–121. ISBN: 0-8186-7940-9. DOI: 10.1109/AST.1997.599921. URL: http://ieeexplore.ieee.org/document/599921/.

Buchenrieder, K. (2000). Rapid prototyping of embedded hardware/software systems. *Proceedings. Ninth International Workshop on Rapid System Prototyping (Cat. No.98TB100237)*. Vol. 5. IEEE Comput. Soc, 2–3. ISBN: 0-8186-8479-8. DOI: 10.1109/IWRSP.1998.676660. URL: http://ieeexplore.ieee.org/document/676660/.

Chong-ho, Y. (2021). *Threats to validity of Research Design*. Accessed February 24, 2021. URL: http://www.creative-wisdom.com/teaching/WBI/threat.shtml.

Cockburn, A. (2007). *Agile software development : the cooperative game*. Eng. 2nd ed. Agile software development series. Place of publication not identified: Addison Wesley. ISBN: 0-321-63007-6.

Coley, G. (2014). *BeagleBone Black System Reference Manual*. BeagleBoard.org.

Embley, D. W. and Thalheim, B. (2011). *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. eng. 1. Aufl. Berlin, Heidelberg: Springer-Verlag. ISBN: 3642158641.

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine. ISBN: 0-599-87118-0.

Fielding, R. T. and Taylor, R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology* 2.2, 115–150. ISSN: 1533-5399. DOI: 10.1145/514183.514185. URL: https://dl.acm.org/doi/10.1145/514183.514185.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials* 17.4, 2347–2376. DOI: 10.1109/COMST.2015.2444095.

Galkin, P. (2016). Analysis models of collection data in wireless sensor networks. *2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)*. IEEE, 233–236. ISBN: 978-1-5090-5715-3. DOI: 10.1109/INFOCOMMST.2016.7905392.

Grönman, J., Rantanen, P., Saari, M., Sillberg, P. and Jaakkola, H. (2018). Lessons learned from developing prototypes for customer complaint validation. *Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*. Vol. 2217, 27–30. ISBN: 9788670314733.

Grönman, J., Sillberg, P., Rantanen, P. and Saari, M. (2019). People Counting in a Public Event—Use Case: Free-to-Ride Bus. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE.

GSMA Intelligence (2020). *IoT connections update: impact of Covid-19 on our forecast*. Accessed December 17, 2020. URL: https://www.gsmaintelligence.com/

product-news/iot-connections-update-impact-of-covid-19-on-our-forecast/.

Gulati, K., Boddu, R. S. K., Kapila, D., Bangare, S. L., Chandnani, N. and Saravanan, G. (2022). A review paper on wireless sensor network techniques in Internet of Things (IoT). *Materials Today: Proceedings* 51, 161–165. ISSN: 22147853. DOI: 10.1016/j.matpr.2021.05.067. URL: https://linkinghub.elsevier.com/retrieve/pii/S2214785321036439.

Hamdan, S., Ayyash, M. and Almajali, S. (Nov. 2020). Edge-Computing Architectures for Internet of Things Applications: A Survey. *Sensors* 20 (22), 6441. ISSN: 1424-8220. DOI: 10.3390/s20226441. URL: https://www.mdpi.com/1424-8220/20/22/6441.

Harjamäki, J., Saari, M., Nurminen, M., Rantanen, P., Soini, J. and Hästbacka, D. (2023). Lessons Learned from Collaborative Prototype Development Between University and Enterprises. *Proceedings of the 33th International Conference on Information Modelling and Knowledge Bases*. Ed. by T. Welzer Družovec, M. Hölbl, L. Nemec Zlatolas and S. Kuhar. University of Maribor, 273–300. DOI: https://doi.org/10.18690/um.feri.5.2023.13. URL: https://press.um.si/index.php/ump/catalog/view/785/1118/3128-2.

Healy, M., Newe, T. and Lewis, E. (2008). Wireless Sensor Node hardware: A review. *2008 IEEE Sensors*. IEEE, 621–624. ISBN: 978-1-4244-2580-8. DOI: 10.1109/ICSENS.2008.4716517. URL: http://ieeexplore.ieee.org/document/4716517/.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly* 28.1, 75–105. DOI: 10.2307/25148625.

HMD Global (2021). *Nokia 7.2*. Accessed February 26, 2021. URL: https://www.nokia.com/phones/en_int/nokia-7-2.

Hunt, A. and Thomas, D. (2000). Addison-Wesley.

IEEE Standard for an Architectural Framework for the Internet of Things (IOT) (2019). New York, USA: The Institute of Electrical and Electronics Engineers, Inc. DOI: 10.1109/IEEESTD.2020.9032420.

Intel Corporation (2014). *Intel Galileo Gen 2 Development Board*. Accessed December 14, 2020. URL: http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-g2-datasheet.html.

International Telecommunication Union (2012). *Overview of the Internet of things (Recommendation ITU-T Y.2060)*.

ISO/IEC/IEEE 42010:2011, Systems and software engineering — Architecture description (2011).

Jaakkola, H., Henno, J. and Thalheim, B. (2016). Why Information Systems Modelling Is Difficult. English. CEUR Workshop Proceedings. Ed. by Z. Budimac, Z. Horvath and T. Kozsik, 29–40.

Jin, X., Chun, S., Jung, J. and Lee, K.-H. (2014). IoT Service Selection Based on Physical Service Model and Absolute Dominance Relationship. *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, 65–72. DOI: 10.1109/SOCA.2014.24. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6978172.

Kernighan, B. W. and Ritchie, D. M. (1978). *The C Programming Language*. 2nd ed. Prentice-Hall, 1–228. ISBN: 9780131101630.

Kitchenham, B. and Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering. Version 2.3. *EBSE Technical Report EBSE-2007-01*.

Kreiner, C., Steger, C., Teiniker, E. and Weiss, R. (2001). A HW/SW codesign framework based on distributed DSP virtual machines. *Proceedings - Euromicro Symposium on Digital Systems Design: Architectures, Methods and Tools, DSD 2001*, 212–219. DOI: 10.1109/DSD.2001.952284.

Kruger, C. P., Abu-Mahfouz, A. M. and Hancke, G. P. (2015). Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components. *2015 IEEE International Conference on Industrial Technology (ICIT)*, 1926–1931. DOI: 10.1109/ICIT.2015.7125378.

Laukkarinen, T. (2015). *Abstracting Application Development for Resource Constrained Wireless Sensor Networks*, 190. ISBN: 9789521535420.

Lee, E. A. and Seshia, S. A. (2017). *Introduction to Embedded Systems. A Cyber-Physical Systems Approach. Second Edition*. Vol. 195, 537. ISBN: 978-0-557-70857-4.

Liou, F. F. (2019). *Rapid Prototyping and Engineering Applications*. 2nd ed. Boca Raton: Taylor & Francis, CRC Press, aa.2009.03329cae.001. ISBN: 9780429029721. DOI: 10.1201/9780429029721.

Mäntyniemi, A., Pikkarainen, M. and Taulavuori, A. (2004). A framework for off-the-shelf software component development and maintenance processes. *VTT Publications* 525, 3–127. ISSN: 12350621.

Marković, D., Vujicic, D., Jovanovic, Z., Pesovic, U., Randik, S. and Jagodic, D. (2016). Concept of IoT System for Monitoring Conditions of Thermal Confort. *International Scientific Conference "UNITECH 2016"*. November. Gabrovo.

Mishra, D. and Mishra, A. (2011). Complex software project development: agile methods adoption. *Journal of Software Maintenance and Evolution: Research and Practice* 23.8, 549–564. ISSN: 1532060X. DOI: 10.1002/smr.528. URL: http://doi.wiley.com/10.1002/smr.528.

Namiot, D. (2015). Time series databases. *Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL2015)* 1536, 132–137. ISSN: 16130073.

Nemeth, E., Snyder, G. and Hein, T. R. (2002). *Linux Administration Handbook*. Prentice Hall PTR, 890.

Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., Rafiq, U., Melegati, J., Guerra, E., Kemell, K.-K., Saari, M., Zhang, Z., Le, H., Quan, T. and Abrahamsson, P. (Oct. 2023). Generative Artificial Intelligence for Software Engineering – A Research Agenda. URL: http://arxiv.org/abs/2310.18648.

Nunamaker, J. F., Chen, M. and Purdin, T. D. M. (1991). Systems Development in Information Systems Research. *Journal of Management Information Systems* 7, 89–106. DOI: 10.1080/07421222.1990.11517898.

Nurminen, M., Lindstedt, A., Saari, M. and Rantanen, P. (2021). The Requirements and Challenges of Visualizing Building Data. *2021 44th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE.

Nurminen, M., Saari, M. and Rantanen, P. (2021). DataSites: a simple solution for providing building data to client devices. *2021 44th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE.

Ojo, M. O., Giordano, S., Procissi, G. and Seitanidis, I. N. (2018). A Review of Low-End, Middle-End, and High-End Iot Devices. *IEEE Access* 6, 70528–70554. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2879615. URL: https://ieeexplore.ieee.org/document/8528362/.

Oracle (2021). *Java*. Accessed February 26, 2021. URL: https://www.java.com/.

Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems* 24.3, 45–77. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302. URL: https://www.tandfonline.com/doi/full/10.2753/MIS0742-1222240302.

Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys Tutorials* 16.1, 414–454. DOI: 10.1109/SURV.2013.042313.00197.

Perkovic, L. (2012). *Introduction to Computing Using Python, An Application Development Focus*. John Wiley & Sons, 484. ISBN: 9780470618462.

Petersen, K., Vakkalanka, S. and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64, 1–18. ISSN: 09505849. DOI: 10.1016/j.infsof.2015.03.007. eprint: arXiv:1011.1669v3. URL: http://dx.doi.org/10.1016/j.infsof.2015.03.007%20http://linkinghub.elsevier.com/retrieve/pii/S0950584915000646.

Porras, J., Khakurel, J., Knutas, A. and Pänkäläinen, J. (2018). Security Challenges and Solutions in theInternet of Things. *Nordic and Baltic Journal of Information and Communications Technologies* 2018 (1), 177–206. ISSN: 1902-097X. DOI: 10.13052/nbjict1902-097X.2018.010. URL: http://www.riverpublishers.com/journal_read_html_article.php?j=NBJICT/2018/1/10.

Python Software Foundation (2021). *Python*. Accessed February 26, 2021. URL: https://www.python.org/.

Raninen, A., Ahonen, J. J., Sihvonen, H.-M., Savolainen, P. and Beecham, S. (2013). LAPPI: A light-weight technique to practical process modeling and improvement target identification. *Journal of Software: Evolution and Process* 25.9, 915–933. ISSN: 20477473. DOI: 10.1002/smr.1571. URL: http://doi.wiley.com/10.1002/smr.1571.

Rantanen, P., Mäkivaara, J., Saari, M., Sillberg, P. and Jaakkola, H. (2021). Utilizing Cost-effective NB-IoT-based Sensors for Detecting Water Temperature and Flow. *25rd IEEE International Conference on Intelligent Engineering Systems 2021*. Submitted. IEEE.

Rantanen, P. and Saari, M. (2020). Towards the utilization of cost-effective off-the-shelf devices for achieving energy savings in existing buildings. *2020 International Conference on Intelligent Systems (IS)*. IEEE.

Raspberry Pi Foundation (2020). *Teach, Learn, and Make with Raspberry Pi*. Accessed December 21, 2020. URL: https://www.raspberrypi.org/.

Rojas, D. and Barrett, J. (2017). A hardware-software WSN platform for machine and structural monitoring. *2017 28th Irish Signals and Systems Conference (ISSC)*. IEEE, 1–6. ISBN: 978-1-5386-1046-6. DOI: 10.1109/ISSC.2017.7983626. URL: http://ieeexplore.ieee.org/document/7983626/.

Roman, R., Lopez, J. and Mambo, M. (Jan. 2018). Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems* 78, 680–698. ISSN: 0167739X. DOI: 10.1016/j.future.2016.11.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X16305635.

Royce, D. W. W. (1970). Managing the Development of large Software Systems. *Ieee Wescon* August, 1–9.

Saari, M., Baharudin, A. M. bin, Sillberg, P., Hyrynsalmi, S. and Yan, W. (2018). LoRa — A survey of recent research trends. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 0872–0877. ISBN: 978-953-233-095-3. DOI: 10.23919/MIPRO.2018.8400161. URL: https://ieeexplore.ieee.org/document/8400161/.

Saari, M., Baharudin, A. M. bin and Hyrynsalmi, S. (2017). Survey of prototyping solutions utilizing Raspberry Pi. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 991–994. ISBN: 978-953-233-090-8. DOI: 10.23919/MIPRO.2017.7973568. URL: http://ieeexplore.ieee.org/document/7973568/.

Saari, M., Baharudin, A. M. bin, Sillberg, P., Rantanen, P. and Soini, J. (2016). Embedded Linux controlled sensor network. *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 1185–1189. DOI: 10.1109/MIPRO.2016.7522319.

Saari, M., Grönman, J., Soini, J., Rantanen, P. and Mäkinen, T. (2020a). Experimenting with Means to Store and Monitor IoT based Measurement Results for Energy Saving. *2020 43rd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE.

Saari, M., Nurminen, M. and Rantanen, P. (2022). Survey of Component-Based Software Engineering within IoT Development. *2022 45th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE.

Saari, M., Rantanen, P. and Hyrynsalmi, S. (2020). Software hardware combination for data gathering. *Proceedings of 2020 IEEE 10th International Conference on Intelligent Systems (IS2020)*.

Saari, M., Rantanen, P., Hyrynsalmi, S. and Hästbacka, D. (2022). Framework and Development Process for IoT Data Gathering. *Advances in Intelligent Systems Research and Innovation*. Ed. by V. Sgurev, V. Jotsov and J. Kacprzyk. (Extension of peer reviewed conference publication Saari, Rantanen and Hyrynsalmi, 2020). Springer International Publishing, 41–60. ISBN: 978-3-030-78124-8. DOI: 10.1007/978-3-030-78124-8_3. URL: https://doi.org/10.1007/978-3-030-78124-8_3.

Saari, M., Sillberg, P., Grönman, J., Kuusisto, M., Rantanen, P., Jaakkola, H. and Henno, J. (2019a). Reducing Energy Consumption with IoT Prototyping. *Acta Polytechnica Hungarica* 16.9, SI, 73–91. ISSN: 1785-8860. DOI: 10.12700/APH.16.9.2019.9.5.

Saari, M., Sillberg, P., Grönman, J., Rantanen, P., Jaakkola, H. and Henno, J. (2019b). Survey of Applications for Apartment Energy Consumption Monitoring. *23rd IEEE International Conference on Intelligent Engineering Systems 2019*. IEEE.

Saari, M., Sillberg, P., Rantanen, P., Soini, J. and Fukai, H. (2015a). Data collector service - practical approach with embedded linux. *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1037–1041. DOI: 10.1109/MIPRO.2015.7160428.

Saari, M., Soini, J., Grönman, J., Rantanen, P., Mäkinen, T. and Sillberg, P. (2020b). Modeling the Software Prototyping Process in a Research Context. *Information Modelling and Knowledge Bases XXXII*. Ed. by M. Tropmann-Frick, B. Thalheim, H. Jaakkola, Y. Kiyoki and N. Yoshida. Vol. 333. IOS Press, 107–118. ISBN: 9781643681405. DOI: 10.3233/FAIA200823. URL: http://ebooks.iospress.nl/doi/10.3233/FAIA200823.

Saari, M., Turunen, J., Linna, P., Aramo-Immonen, H. and Huhtala, M. (2015b). Explorative study of teaching programming to vocational teachers in Finland.

7th International Conference on Education and New Learning Technologies, 2860–2869.

Saha, D., Mitra, R. and Basu, A. (1997). Hardware software partitioning using genetic algorithm. *Proceedings Tenth International Conference on VLSI Design*. IEEE Comput. Soc. Press, 155–160. ISBN: 0-8186-7755-4. DOI: 10.1109/ICVD.1997.568069. URL: http://ieeexplore.ieee.org/document/568069/.

Scacchi, W. (2002). Process Models in Software Engineering. *Encyclopedia of Software Engineering*. Hoboken, NJ, USA: John Wiley & Sons, Inc. DOI: 10.1002/0471028959.sof250. URL: http://doi.wiley.com/10.1002/0471028959.sof250.

Semtech S.A (2021). *Lora network packet forwarder project*. Accessed March 2, 2021. URL: https://github.com/Lora-net/packet_forwarder.

Shadish, W., Cook, T., Campbell, T. (2002). *Experiments and generalized causal inference*. Vol. 100. 470, 1–81. ISBN: 0395615569. URL: http://impact.cgiar.org/pdf/147.pdf.

Sillberg, P., Gronman, J., Rantanen, P., Saari, M. and Kuusisto, M. (2018). Challenges in the Interpretation of Crowdsourced Road Condition Data. *2018 International Conference on Intelligent Systems (IS)*. IEEE, 215–221. ISBN: 978-1-5386-7097-2. DOI: 10.1109/IS.2018.8710571. URL: https://ieeexplore.ieee.org/document/8710571/.

Sillberg, P., Rantanen, P., Saari, M., Leppäniemi, J., Soini, J. and Jaakkola, H. (2009). Towards an IP-based alert message delivery system. *ISCRAM 2009 - 6th International Conference on Information Systems for Crisis Response and Management: Boundary Spanning Initiatives and New Perspectives*. Ed. by J. Landgren and S. Jul. June 2015. Information Systems for Crisis Response and Management, ISCRAM, 8 p. ISBN: 978-91-633-4715-3.

Sillberg, P., Saari, M., Grönman, J., Rantanen, P. and Kuusisto, M. (2020). Interpretation, Modeling and Visualization of Crowdsourced Road Condition Data. *Intelligent Systems: Theory, Research and Innovation in Applications*. Ed. by R. Jardim-Goncalves, V. Sgurev, V. Jotsov and J. Kacprzyk. (Extension of peer reviewed conference publication Sillberg, Gronman, Rantanen, Saari and Kuusisto, 2018). Springer International Publishing, 99–119. ISBN: 978-3-030-38704-4. DOI: 10.1007/978-3-030-38704-4_5. URL: https://doi.org/10.1007/978-3-030-38704-4_5.

Soini, J., Kuusisto, M., Rantanen, P., Saari, M. and Sillberg, P. (2019). A Study on an Evolution of a Data Collection System for Knowledge Representation. *Information Modelling and Knowledge Bases XXXI*. Ed. by A. Dahanayake, J. Huiskonen and Y. Kiyoki. Vol. 321. IOS Press, 161–174. DOI: 10.3233/FAIA200013.

Sommerville, I. (2016). *Software Engineering (10th edition)*. 10th ed. ISBN: 978-1-292-09613-1.

Srivastava, M. and Brodersen, R. (1991). Rapid-prototyping of hardware and software in a unified framework. *1991 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*. IEEE Comput. Soc. Press, 152–155. ISBN: 0-8186-2157-5. DOI: 10.1109/ICCAD.1991.185217. URL: http://ieeexplore.ieee.org/document/185217/.

Stroustrup, B. (2013). *The C ++ Programming Language*. Fourth edi. Addison-Wesley, 1366. ISBN: 9780321563842.

The Industrial Internet of Things Volume G1: Reference Architecture, v1.9 (2019). Industrial Internet Consortium, a program of Object Management Group, Inc. URL: https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf.

Vakaloudis, A. and O'Leary, C. (2019). A framework for rapid integration of IoT Systems with industrial environments. *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 601–605. ISBN: 978-1-5386-4980-0. DOI: 10.1109/WF-IoT.2019.8767224. URL: https://ieeexplore.ieee.org/document/8767224/.

Weiser, M. (1991). The Computer for the 21 st Century. *Scientific American* 265.3, 94–105.

Xu, L. D., He, W. and Li, S. (2014). Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics* 10.4, 2233–2243. ISSN: 1551-3203. DOI: 10.1109/TII.2014.2300753. URL: http://ieeexplore.ieee.org/document/6714496/.

PUBLICATIONS

PUBLICATION

I

Data collector service - practical approach with embedded linux

Saari, M., Sillberg, P., Rantanen, P., Soini, J. and Fukai, H.

*38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*2015, 1037–1041
DOI: 10.1109/MIPRO.2015.7160428

Data Collector Service – Practical Approach with Embedded Linux

M. Saari*, P. Sillberg*, P. Rantanen*, J. Soini* and H. Fukai**
* Tampere University of Technology/Department of Software Engineering, Pori, Finland
** Keio University/Faculty of Policy Management, Fujisawa-shi, Japan
mika.saari@tut.fi

Abstract - Nowadays embedded systems are one of the most important application areas in information technology. Embedded systems are often used in life critical situations, where reliability and safety are more important criteria than performance. This paper presents a data collector service that has been developed based on embedded Linux, which operates as a key element in a larger intelligent alarm system. The target of this study was to test out how well a cost-efficient single-board computer could be used to gather sensory data, and how this data can be provided for the client over the public Internet. The paper describes the data collector service currently in use and its functionality and also gives a concrete example of how to utilize a microcontroller with an embedded Linux distribution. The paper presents one solution on how to utilize embedded systems for managing and controlling conditions in buildings and also environmental conditions in a smart and cost-effective way.

I. INTRODUCTION

Sensors are commonly utilized components including various kinds of warning and alarm systems. With the huge development of sensor technology, it has been possible to create minuscule, reasonably priced components, sensors and controllers with ultra-low power consumption. These kind of components are enabled for the rapidly improvement of the sensor networks and therefore it can be seen how sensor network implementations have been applied in numerous different fields of operation. A sensor network consists of single sensors, with the purpose of sensing the surroundings and to forward the collected data. There is a great variety of sensors available and their classification is based mainly on the features of the information collected. These kinds of physical features include temperature, humidity, brightness or air pressure.

In this research, one particular subject was to sense the condition changes in indoor spaces and data collection and entry related to their authentication as well as transmitting them forward. The objective of the research was to construct an automatic service – i.e. a data collector service – for collecting condition data that would, in turn, enable the collection and transmission of condition data for a backup system that exploits and analyzes measurement data as flexibly as possible in real-time. The starting point was the development of a system optimally attending to an independent, selected backup system for collecting, recording and transmitting condition data. The aim was to study and develop a solution that was as simple, reliable, cost-effective and easy to maintain as possible for the defined purpose. In this case, it was decided to implement the solution by adapting a microcontroller with embedded Linux for data collection and distribution. Embedded systems usually adopt embedded Linux as the operating system because of the numerous economic and technical benefits – the Linux kernel sources are well structured so that CPU-specific code is easy to find and is minimized. A prototype for realizing the service in question, which will be presented in detail in this paper, was created during the KiiauData research project in 2014.

The study presented here is part of the intensive collaboration between the Tampere University of Technology (TUT) in Finland and Keio University in Japan. The Global Environmental System Leaders (GESL) Program, ongoing in Keio University's Shonan Fujisawa Campus (SFC), and the Alert system for detecting anomalous situations developed as one of its outcomes serve as the background of the research. One main part of the alert system under development is the *data collector service* which was developed in collaboration by the university partners in TUT Pori, enabled by the ongoing KiiauData (Smart analysis of property systems data) project. One of the main aims of this two-year (2013-2014) project, funded by TEKES [1], was to study potential new technologies for managing and controlling conditions in buildings in a smart way. The expected results of the project will enable providers of products and services in the built environment to form wider and more automated solutions both for new breakthroughs and recognized problems in the smart built environment. The data collector service presented here is one concrete example of the studied and piloted solutions produced during the joint project.

Related research in this specific area – i.e. utilization of a microcontroller with embedded systems – has been conducted by Rakesh et al. [2], for example, who have introduced a system which implements an embedded system for monitoring wireless sensor nodes and a camera installed inside a building for security surveillance. Toshniwal and Conrad [3] have studied how to make a cost-effective network-based sensor monitoring system which is portable for various applications. They have developed Linux-based systems based on desktop architecture with a sensor package, and also another system which used an embedded Single Board Computer

(SBC) together with sensors. In addition, Cheng and Shen [4] have introduced a wireless sensor network communication terminal based on embedded Linux. Voinescu et al. [5] describe a device which can work as a network connection to a single board computer (BeagleBone or similar), where the target was to make an easy-to-use wireless networking device. Sawant et al. [6] studied a device that is capable of making file manager operations with two USB flash drives. Their study gives basic knowledge of using of an ARM-based embedded Linux and touch screen. Banerjee et al. [7] proposed and implemented the design of a secure sensor node prototype. They built the prototype using a single board computer (Raspberry Pi in this case), accelerometer, and Bluetooth dongle. The above-mentioned studies deal with the same research area and have a very close connection to the specific research topic presented in this paper.

The following section (Section 2) briefly describes the background system and Section 3 gives a detailed explanation of how its first part – the data collector service – has been carried out. Section 4 includes a discussion and suggestions for future research on the topic and finally section 5 summarizes the study.

II. BACKGROUND – THE INTELLIGENT ALERT SYSTEM

The basis of the study is the alert system under development in Keio SFC. With this planned system it is possible to collect environmental data for different purposes. The system will make environmental sensing easier in various places by using small sensors, and there are many ways to utilize this collected data. One of the intended applications for using the system is for detecting anomalous situations. The aim of such a system is to handle environmental sensor data collected from multiple locations. In practice, it is not easy to understand the problems inherent in a given place just by looking over a graph of sensor data. Therefore, an alert system is need for interpreting the environmental changes in the space in question and associated problems.

This intelligent alert system consists of three (3) main parts shown in Fig. 1: *the first part* is the collection of environmental sensor data via the new *data collector service. The second part* is the detection of anomalous situations by utilizing the sensor data. *The third part* is sending the alert to where the situations were detected.

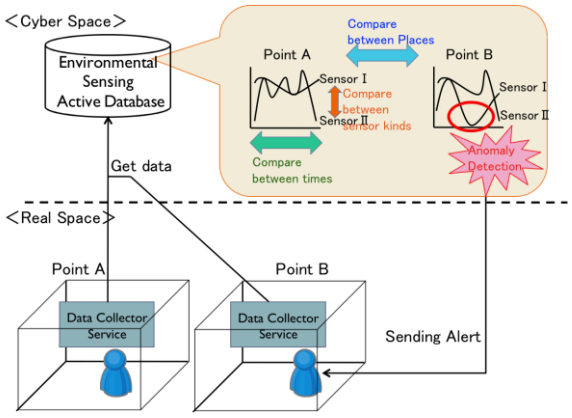

Figure 1. Overview of the Alert system

This particular system targets indoor spaces utilized by the public such as offices, meeting rooms, stations, trains, etc. In this case, the variables being sensed include temperature, light, and humidity. The system automatically collects the sensor data from sensors placed in various places, and analyzes changes over time. As environmental sensor data is collected for a given place over a long time-span, the results become more useful, not only by evaluating the results at one location but also by comparing the results from the sensor data from many locations. Each type of sensor data has four (4) feature values: location information, time, data type, and sensing value. The alert system registers the collected sensing data with an *active database* system in real time. The active database (upper left corner in Fig. 1) automatically reacts in response to detected state change rules that have been pre-defined by the users of the alert system. If an anomalous situation is detected as specified in the active database's rule set, the database system *sends alerts* to the anomalous location. In this case, the role of the active database is to support rule definition, compare data between sensors, compare similar data at different time-points, and compare between different sensor types.

This paper deals with the first part of the alert system, i.e. the *data collector service* (lower left corner in Fig. 1), and does not describe the Intelligent Alert System as a whole. The paper presents the features and architecture, hardware and software components of the developed service, and also the physical connections between the components. The structure of the developed software used for data collection is also described.

III. DATA COLLECTOR SERVICE – IMPLEMENTATION

The architecture of the *Data Collector Service* is shown in Fig. 2. The purpose of the service is to measure light, temperature, and humidity. The sensors used are shown in Fig. 2 (right side). The data can be collected continuously, up to six times a minute. The collected data is then provided as a service for clients, who access the service using the Internet (left side of Fig. 2). In our use case the clients can also be other service providers.

The physical device itself is placed in a public space – TUT's laboratory in this case – to collect data. Attention had to be paid to the physical size of the device as a device smaller than an ordinary PC is easier to install in a public area. In this implementation, the device must also have Ethernet or Wi-Fi capabilities in order to be remotely accessible. These two requirements lead to the use of a device with embedded Linux. The embedded Linux in this context means Linux that can run on ARM-based processors. The embedded Linux devices often have database and web server capabilities, or they can be easily added afterwards. The size of the device's internal mass memory is not critical as long as the device has peripheral ports for external flash memory or Secure Digital (SD) cards.

The integral component of the Data Collector Service, the BeagleBone Black [8], is a low-cost, high-expansion focused SBC using an ARM Cortex-A8 based processor. It can host a Linux operating system and has a 10/100 Ethernet connection and a microSD connector, with 512

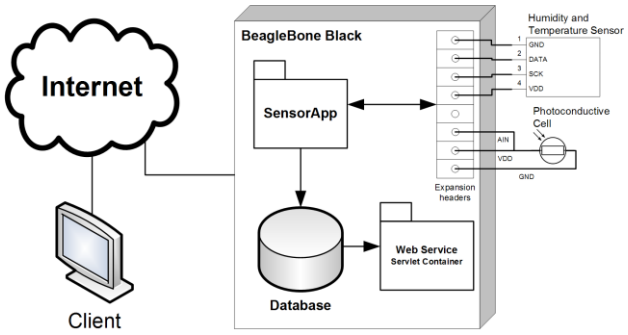

Figure 2. System Architecture of Data Collector Service

MB system memory and 2 GB of embedded MultiMediaCard (eMMC) memory.

The BeagleBone Black has two expansion headers, labeled P8 and P9, which allows the integration of BeagleBone electronics projects [8, 9], and in this research the features of these expansion headers were used. The sub system developed utilizes GND (Ground), 3.3V and 1.8V power, two GPIO (General Purpose Input/Output), and AIN (analog input) pins to drive light, humidity, and temperature sensors.

A. BeagleBone Black with Embedded Linux

The BeagleBone board comes with a pre-installed operating system called Ångström Linux**,** which is a stable and user-friendly distribution for embedded devices and is categorized in the Embedded Linux category. [10] Despite being designed for embedded devices, the Ångström Linux has many of the capabilities that can be found in other full-fledged Linux distributions, such as the X11 windowing system and a substantial amount of software packages in its package repositories.

Some basic Linux server hardening configurations were made to the Ångström Linux for more secure network operation. For example, the system time management was changed to use ntpdate [11], the X11 service was disabled, direct root user access was removed and a new basic user was created which could be used for remote Secure Shell (SSH) access. Python programming language was used together with an Adafruit-BeagleBone-IO-Python library, to utilize the I/O operations of the sensors [12].

The Data Collector Service itself does not utilize the data it collects, as this was to be done by a remote computer with more computing capacity. Thus the Data Collector Service only collects the data and serves it over the Internet for use or as input to the next part of the alert system. We chose to store the data to persistent memory so it could be later accessed by one or more clients. In this case there were no special requirements on how, when and how often the data should be delivered to the remote computer, so the decision was to implement relatively simple server software utilizing the client-server architecture. This way the consumer of the sensor data can decide the most convenient update cycle.

The Data Collector Service provides the data to the clients over a representational state transfer (REST) HTTP/GET [13] interface. Clients may access the interface for historical data from a chosen time interval, or if they choose to, poll periodically for the newest data. By using a short enough polling interval, it would be possible to get near real-time data from the service within the limitations of the computing performance of the BeagleBone platform. The number of new data points per minute would be limited by how often the sensors are read on the Data Collector Service.

The server software was deployed on an Apache Tomcat web server [14] and the sensor readings stored in a MySQL database [15]. In Fig. 2, these two components are called *Web Service* and *Database*, respectively. A third major software component called *SensorApp* also runs on the *BeagleBone Black* platform. The task of the *SensorApp* is to communicate with the *Expansion headers* which are used to drive the physical sensors attached to the BeagleBone Black.

The *SensorApp* and the *Web Service* running on top of the servlet container were written by the project team. The *Database* and *Servlet Container* were written by a third party. Ångström's own package manager provided MySQL, while Apache Tomcat and Java Virtual Machine (Oracle's Java in this case) were installed using their latest available installation packages for Linux operating systems.

Fig. 2 also illustrates the directions of data flow which happen between the components. The *Web Service* can only read data from the *Database*, while the *SensorApp* has read and write access to the device's GPIO headers in order to operate the sensors, and it also directly writes the collected data into the *Database*. For the completeness of the REST interface, the full CRUD (Create, read, update and delete) operations could have been implemented through the interface, but that would have increased the total code complexity of the system. In addition, it would have raised security issues, such as the malicious removal or modification of the collected data. Unauthorized access could be mitigated by the use of access control such as passwords or certificates. In the end, the prime interest was on collecting the data, so keeping outside access as "read only" was the most effective method in terms of computational capacity and code complexity.

Fig. 3 shows the architecture of the physical device with the sensors attached. BeagleBone expansion header

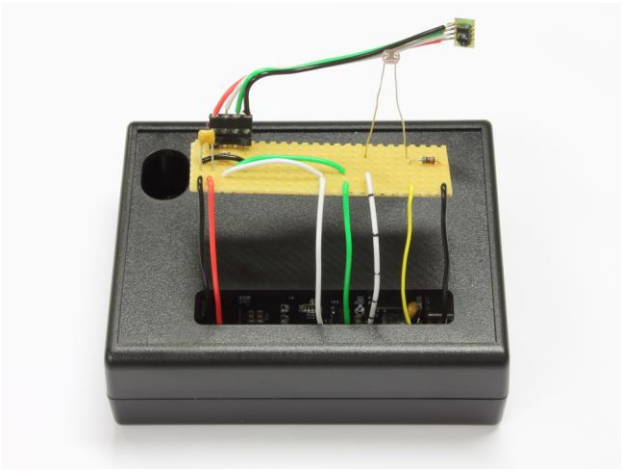

Figure 3. Sensor device used for Data Collector Service

P9 was used to connect the sensors – a photoconductive cell (NSL-19M51 [16]) and humidity and temperature sensor (SHT11 [17]). The photoconductive cell is connected by using a typical application circuit [18]. The humidity and temperature sensor is connected to the BeagleBone by using a datasheet application circuit [17].

B. System Functionality

The web service provides the collected data through one read-only interface. The interface can be accessed by using a simple HTTP GET request. The default query without any parameters returns the ten latest data points. By using different parameters (such as begin_date, end_date, limit and paging) one can request a desired data set from the service. Fig. 4 shows the sequence of accessing the data from the web service. The *Client* connects to the device's *Web Service* interface, which in turn retrieves the requested data from the *Database*. The data is then marshaled into XML format by utilizing JAXB (Java Architecture for XML Binding) annotations and sent back to the client.

As the data is stored to the device's persistent memory, the data can be requested when needed and as often as needed. However, due to the limited computing resources of the platform, a request for a large data set may take a long time or even fail. The code for the web service was not specially written nor optimized for this use case, but merely as a generic proof-of-concept implementation. The marshaling of the XML output at least could have been done in a more memory-efficient way, or a different approach such as JSON serializing could have been used instead. An often-used paradigm is to limit the maximum measurement count to a known safe figure, and use the paging parameter to retrieve the rest of the results. The current implementation appears to cap at around 40000 measurements (or about five days of collected data) on the device, while a desktop machine was able to double that amount (both are with the default Java VM settings). Requesting any more than the aforementioned number of measurements either causes serious degradation of performance, or results in an out-of-memory situation because of the limited memory capabilities of the BeagleBone platform.

The sensor data is collected in a separate process to the web service. Fig. 5 is an illustration of one loop of the sensor reading process. Each loop corresponds to a single measurement point. The humidity and temperature sensor used has its own built-in circuit, and the measurements can take up to 80 or 320 milliseconds at the default accuracy (12 bits for humidity and 14 bits for

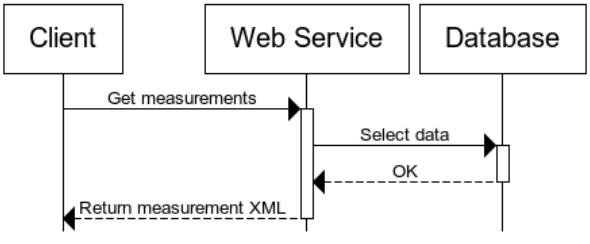

Figure 4. Sequence diagram for accessing the data from the web service

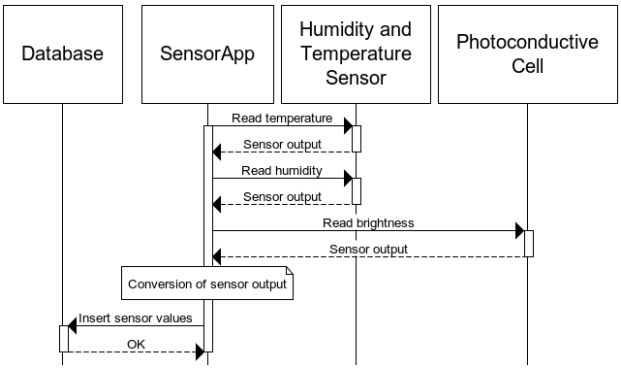

Figure 5. Sensor device used for Data Collector Service

temperature). Also, in order to avoid excess self-heating of the sensor, a maximum of one measurement per second at 12 bit accuracy should be made. [17] These limitations set a theoretical maximum of 15 measurements per minute at default accuracy (when the communication with the sensor is not taken into account). When a lower accuracy (20ms/8bit and 80ms/12bit) is used, a maximum of 60 measurements per minute can be achieved. For this application, it was decided to use the default accuracy and an interval of six measurements per minute.

As can be seen in Fig. 5, first the application reads the raw temperature value for the combined *humidity and temperature sensor*, and then pauses for a while to let the sensor cool down, after which the raw humidity value is read from the same sensor. Then the raw value from the *photoconductive cell* is read. The raw signal values have to be converted before they can be stored to the *database* by using the conversion formulas provided by the manufacturer. After the database has been updated, the application enters sleep mode to attain the desired measurement interval.

IV. DISCUSSION & FUTURE RESEARCH

This paper presents a *data collector service*, which utilizes BeagleBone Black development board with an embedded Linux distribution. The goal was to experiment how well a cost-efficient SBC can be used to gather sensory data, and how this data can be provided to the client over the public Internet. This goal was reached successfully, and the designed system was tested and found to work as planned. Nevertheless, the development process raised several improvement ideas, which could be realized in the future.

One of the issues is the packaging of the sensor system. The current version was a prototype version, and consequently the focus was on making the system functional, both by testing the sensor connections and readings, as well as benchmarking the functionality of the REST API and the software components. This limited the practical usability of the system, for example, making it unfit for use outdoors. In addition, the current software consists of various libraries, programming languages and components, and is as such slightly tricky to install. The software components could be packaged into a single application for easier installation. There is also the possibility to release the source code as an open source release.

The chosen sensor components could also be improved. The serial interface of SHT11 offers good power efficiency, but it cannot be addressed by standard I2C (Inter-Integrated Circuit) protocol, which would make programming tasks easier. In this use case the system is always provided with a continuous power supply and thus better programmability would be a major asset in future studies. In our case, the limited amount of sensor devices to be deployed makes the cost of the single board computer largely irrelevant, though it should be noted that there are other SBCs that are slightly cheaper, but still offer reasonable computing performance. One popular choice is the RaspberryPi (e.g. [19]), which would work – specification-wise – equally well in this use case. Both of these boards offer excellent extension capabilities and can be expanded with additional sensors. The addition of multiple sensors raises another issue, which has not been studied in this research. It is unclear how well the board and the developed system would cope with a very large number of sensors. Also, in this system, all sensors are located very close to the actual board, and thus, the signal degradation can be thought negligible, but this is not necessarily true in large-scale monitoring systems. One example of this kind of system would be the monitoring of an entire apartment complex, where sensors are physically located very far apart and connected to the board by long wires or cables. In this case, it would be possible to deploy a multiple sensor system, but simply using multiple sensors with a single board is a more cost-efficient solution.

In addition to the aforementioned improvement ideas, our future research will focus on topics only briefly discussed in this paper, such as the utilization of the collected data using the designed REST API in various end-user applications.

As can be seen, there are many ways to continue and improve this study. However, the current service produces real-time data regularly and reliably for the benefit of the main system. The prototype developed has proven to be stable and reliable in practice. Work on building the final alarm system is currently ongoing at Keio University.

V. SUMMARY

The paper introduced a prototype system created for sensor data collection and transmission. The presented data collector service is part of a larger alarm system and provides sensor data for the main system. The aim of the study was to test how a single board computer can be used to gather sensory data and how this data can be provided to clients over the public Internet. The paper presented the features and architecture of the developed service, the used hardware and software components, the physical connections between the components, and also the structure of the software. The paper gives a concrete example of how to utilize a microcontroller with an embedded Linux distribution.

REFERENCES

[1] Finnish Funding Agency for Technology and Innovation, Tekes, http://www.tekes.fi/en. Retrieved February 6th, 2015.

[2] V. S. Rakesh, P. R. Sreesh, and S. N. George, “An improved real-time surveillance system for home security system using BeagleBoard SBC, Zigbee and FTP webserver,” In India Conference (INDICON), 2012 Annual IEEE, pp. 1240–1244, December 2012.

[3] K. Toshniwal, and J. M. Conrad, “A web-based sensor monitoring system on a Linux-based single board computer platform,” Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon), pp. 371–374, March 2010.

[4] X. Cheng, and F. Shen, “Design of the wireless sensor network communication terminal based on embedded Linux,” 2011 IEEE 2nd International Conference on Software Engineering and Service Science, pp. 598–601, July 2011.

[5] A. Voinescu, D. Tudose, and D. Dragomir, “A lightweight, versatile gateway platform for wireless sensor networks,” In Networking in Education and Research, RoEduNet International Conference 12th Edition, pp. 1–4, September 2013.

[6] T. Sawant, B. Parekh, and N. Shah, “Computer independent USB to USB data transfer bridge,” 6th International Conference on Emerging Trends in Engineering and Technology, pp. 40–45, December 2013.

[7] S. Banerjee, D. Sethia, T. Mittal, U. Arora, and A. Chauhan, “Secure sensor node with Raspberry Pi,” Impact-2013, pp. 26–30, November 2013.

[8] G. Coley, BeagleBone Black System Reference Manual. 2014.

[9] M. Richardson, Getting Started with BeagleBone. Sebastopol, CA: Maker Media, 2014.

[10] L. Merciadri, and K. Koen, Angstrom Manual. 2010.

[11] D. L. Mills, ntpdate - set the date and time via NTP, http://doc.ntp.org/4.2.6p5/ntpdate.html. Retrieved February 6th, 2015.

[12] J. Cooper, “Setting up IO Python library on BeagleBone Black,” https://learn.adafruit.com/downloads/pdf/setting-up-io-python-library-on-beaglebone-black.pdf. Retrieved February 6th, 2015.

[13] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

[14] The Apache Software Foundation, Apache Tomcat, http://tomcat.apache.org. Retrieved February 6th, 2015.

[15] Oracle Corporation, MySQL, http://www.mysql.com. Retrieved February 6th 2015.

[16] L. Curvan, Data Sheet NSL-19M51, TO-18 Open Plastic Encapsulated. 2008.

[17] SENSIRION AG, Datasheet SHT1x. 2011.

[18] PerkinElmer Optoelectronics, Photoconductive Cells and Analog Optoisolators (Vactrols®). 2001.

[19] V. Vujović, and M. Maksimović, “Raspberry Pi as a wireless sensor node: performances and constraints,” 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) , pp. 1247–1252, 2014.

PUBLICATION

II

Embedded Linux controlled sensor network

Saari, M., Baharudin, A. M. bin, Sillberg, P., Rantanen, P. and Soini, J.

*39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*2016, 1185–1189
DOI: 10.1109/MIPRO.2016.7522319

Embedded Linux Controlled Sensor Network

M. Saari [*], A.M. Baharudin [**], P. Sillberg [*], P. Rantanen [*] and J. Soini[*]
[*] Tampere University of Technology/Pori Department, Pori, Finland
[**] Keio University, Tokyo, Japan
mika.saari@tut.fi

Abstract - This study utilizes a simple model for constructing sensor nodes – master controller combinations in the Internet of Things. The model combines hardware and software for embedded systems which measure a predefined set of parameters. The master controller manages several sensor nodes, collects data from them and provides data for clients. The paper introduces a proof-of-concept implementation based on the model. The implementation uses an embedded Linux based small computer and microcontroller based sensor nodes in the context of condition measurement, and represents a way to use wireless data transfer between controller and nodes. The target of this study was to test the model, to determine how well a cost-efficient single-board computer could be used to gather sensory data from several sensor nodes, and how this data can be provided for clients over the public Internet.

I. INTRODUCTION

Wireless sensor networks have developed at a fast pace in recent years and have also been one of the major focuses of research in wireless technology. This rapid development has been facilitated by the evolution of electronics miniaturization, growth in performance and energy efficiency, and the development of protocols. Through the fast performance, optimization and miniaturization technology of hardware, the sensors that collect environmental information from the surroundings have been miniaturized. The developments of ever smaller processors and falling prices have enabled brand-new uses for electronics.

Embedded systems are typically designed for a specific application or purpose and come in a variety of shapes and sizes. Moreover, embedded systems are suitable for application systems with strict requirements for functionality, reliability, cost, size, and power consumption. One application of embedded systems that is in constant demand and under continuous development is telecommunications components. The price of wireless components in particular has fallen so much that experts can see clear potential there. In regards to rising energy costs and energy savings, regulation and control technology for buildings would seem to offer considerable potential for embedded solutions. In addition, the property, safety and surveillance technology sector offers growth in the area of embedded systems.

Because of the numerous economic and technical benefits, embedded systems usually adopt embedded Linux as the operating system. This paper introduces a proof-of-concept implementation that uses a small embedded Linux based computer and microcontroller based sensor nodes in the context of condition measurement, and represents a way to use wireless data transfer between controller and nodes. The paper introduces the implementation of a sensor network solution for sensor data collection and transmission. This study was performed in intensive collaboration between Tampere University of Technology (TUT) in Finland and Keio University in Japan. The implementation was created in the TUT Pori department in 2015.

The construction of this paper is as follows. In section II, we review related work. The model of our system is introduced in Section III. In section IV we describe the proof-of-concept implementation. Section V includes a discussion and suggestions for future research on the topic and finally, section VI summarizes the study.

II. BACKGROUND

The Internet of Things (IoT) is the expansion of Internet services, which connects everyday physical objects to the network. This connection between network and physical objects makes it possible to access remote sensor data and to control the physical world from a distance. The first mention of the term IoT is said to have come from Kevin Ashton in 1999. There are also several books about IoT. The Amazon web store found 3114 instances with the phrase “Internet of Things” in January 2016. In this paper we introduce one implementation of the IoT-world.

In our earlier research [1], the focus was on collecting data from one sensor packet which was connected by wire directly to a single-board computer. This was done by using embedded Linux and BeagleBone Black hardware, which is a credit card–sized single-board computer similar to Raspberry Pi. The focus of the research was to collect data and to deliver it over the network. The plan was to build several of these data collector service computers and use them in a specific alarm system.

In this research the focus has been redirected toward the Wireless Sensor Network (WSN) type of solution. A survey conducted in 2002 compiled the basic features of sensor networks [2]. The aim was to collect data from several points to one master node. The collected data are provided to the network. This data could be used in smart house type construction. For example [3] presented a small smart house system, based on one Arduino development board. In that research, the proposed system monitored the environment and also controlled lights, temperature, alarms, and other household appliances. In Finland, where heating is necessary most of the year, a

low cost sensor network could be used for monitoring and controlling the heating and air ventilation systems.

The research started with modeling the construction of a system with one master and several nodes. We introduced an abstract model of the sensor network. There are several studies about more complex models designed for IoT. For example, one research study [4] introduced the Physical Service model, where they described a device model, resource model, and service model. Another piece of research [5] introduced a Wireless Sensor network abstraction model that has three levels: node abstraction, network abstraction and infrastructure abstraction. If this is compared to our research, we focused on the simpler infrastructure abstraction and we used low-cost off-the-shelf equipment available from a local store for our example implementation. The choice of hardware supports rapid prototyping and the test configurations are easily repeatable. For example, [6] and [7] use the Raspberry Pi kind of approach in their wireless sensor network gateway prototyping. The model itself does no set any limitations but one target was to use low-cost and easy-to-use hardware. Also, open hardware and open source software were the selection criteria for the components of the implementation.

The security issues are an important part of wireless sensor networks – How to prevent information leakage or whether the transferred data is vulnerable. The research about security issues of wireless data communication was made by [8]. They focused on ZigBee [9], which we also used in the proof-of-concept implementation.

III. Model of the Sensor Network

The model of the sensor network is shown in Fig. 1. The s*ensor nodes* are used to gather measurements such as temperature, humidity and air pressure from the environment. The number of sensor nodes is not limited to any particular amount and in theory there can be an unlimited number of nodes. The nodes are meant for simple tasks which consist only of passive data collection without the need for advanced data analysis or preprocessing. The *Master node*, in general, has the capability to run a full-feature operating system, and its job is to control the sensor nodes and manage the data collection process. The collected data is stored on the master and provided for client devices over the public Internet or using a more restricted local network.

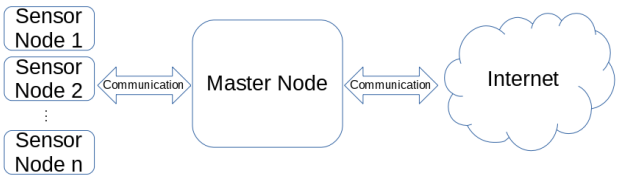

Figure 1 Model of sensor network.

In an earlier research study, we concluded that when using cost-effective single-board computers, such as the BeagleBone, the limited computing performance could cause issues [1]. Problems arise when a single computer (i.e. the master node) runs the web service and database, and also measures the sensor data, in which case the requirements for the hardware can simply be too much. To mitigate this issue, separate sensor nodes are added, which manage the actual process of collecting the data. This way the number of sensors can be increased without additional strain on the limited processing capabilities of the master node.

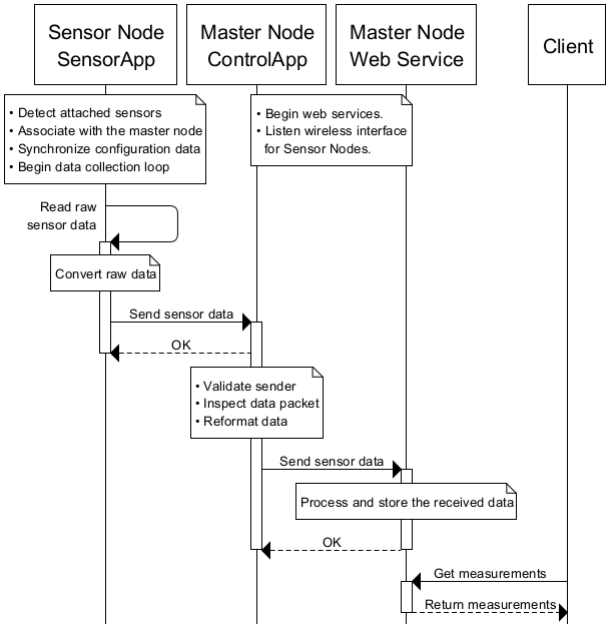

Figure 2 Sequence of operations.

The collection, storing, providing and usage of the sensor data are carried out by four components – or roles. Fig. 2 shows the sequence of operations of each of the components. The *Client* is the consumer of the collected data. The client retrieves the data by accessing the *Master Node*, which consists of two software components to facilitate the collection, the storage and delivery of the sensor data. The master node is a centralized gateway for the *Sensor Nodes*. They include the software that collects and relays the sensor data to the master node. In our prototype system, the client is simply a web browser accessing the data for visualization purposes. The rest of this section describes the details of the model and the three other components – SensorApp, ControlApp, and Web Service.

A. Sensor Nodes

The sensor nodes are devices that provide the raw data. They should be inexpensive, easy to deploy and replaceable. Each sensor node may have a different sensor configuration, and they should operate independently from other sensor nodes.

It was determined that the sensor node should implement the following features (steps denoted in *italics* are not part of the prototype implementation):

1) *Detect attached sensors.*
2) *Associate with the master node and synchronize configuration data.*
3) Read raw data from the attached sensors.
4) Convert the raw data to a transferable format.

5) Send the data over a wireless network to the master node.

6) Sleep and start over from the third step.

The choice of data format used between the sensor nodes and the master node depends on: the available network, bandwidth, and computational resources; developer preference; and use case. Especially for testing and debugging, a human readable format is recommended, but not strictly required.

B. Master Node

The requirements for the master node are higher than for a sensor node. The master node is specified to be a gateway between data consumers (clients) and data producers (sensor nodes). The master node should be powerful enough to execute multiple processes, such as:

- communications with sensor nodes,
- database operations and
- web services.

As the master node is powered on, it will run the code to listen for sensor nodes to associate and send the data. It will also start the *Web Service* for processing the data and delivering it to the outside world over the Internet.

When the master node receives the data packet, it will inspect the address and details of the sender, compare it to the associated sensor nodes, and parse the data. As a security measure, the master node could be instructed to discard all data sent by an unknown sensor node or by a sensor node that does not provide the correct configuration.

If the data received from sensor nodes is not in a structured format it should be transformed to allow easier use. Commonly used formats for Internet applications are JSON and XML, and both are good choices for delivery of data to clients.

IV. EXAMPLE SYSTEM IMPLEMENTATION

The example works as a proof-of-concept implementation, and we do not have a particular use case for it as such. In our earlier publication [1], we presented a system for environment sensing, which utilizes various sensors for collecting measurements (e.g. temperature, humidity), and the case is still equally valid. In fact, the model presented in this paper can be seen as an improvement on the system described, and the model has also been developed based on the findings of the studies performed earlier. The primary purpose of the prototype solution was to show that a feasible system for remotely collecting sensor data can be constructed based on the model.

For the master node we chose an Intel Galileo Gen 2 Development Board [10], which is based on Intel x86 architecture. The Galileo is a single-board computer similar to Beagle Bone or Raspberry Pi. Each of the three boards has the features required to implement our example use case, and in principle, either of the other boards could have been chosen instead. We chose Arduino Uno for the sensor nodes. Galileo, BeagleBone and Raspberry Pi belong to a higher price category than Arduino Uno. Because of the lower price, Arduino Uno makes a more feasible platform for numerous sensor nodes.

Galileo includes a 10/100 Ethernet connection, which we use to connect to the public Internet for the purpose of delivering the collected data to the clients. The web services are provided by a server built on Node.js [11]. The Node.js instance can be somewhat resource intensive, but it seemed to work acceptably in our tests with a small number of concurrent users.

The internal memory of the Galileo is quite limited, so we installed the Yocto Linux operation system [12] on a microSD card to provide a larger storage space. This is especially important, as the database is also located on the master node, and the built-in memory may not be capable of holding all of the collected data. Additionally, the embedded Linux based operation systems usually include common Linux software such as Secure Shell (SSH) server for easier configuration of the node and they are also capable of running other applications primarily targeted for full-feature desktop or server computers (e.g. Apache Tomcat, Java virtual machine).

One advantage of the Galileo board is the support for ready-made hardware expansions shields designed for Arduino. In our example implementation we used the less powerful Arduino Uno [13] for the sensor nodes, which allows us to use the same expansion components for both the master node and the sensor nodes. In this construction we used the Arduino Wireless Proto Shield with XBee modules [14] to implement the communication between the master node and the sensor nodes. The XBee modules are based on ZigBee and are designed for low-power wireless networks. The expansion modules make it possible to add and remove components easily reducing the need for soldering.

Communication with XBee can be done using the basic AT or the more advanced API mode [15]. In our implementation, the XBee API mode was utilized. Using the API mode with packet communication allows the transport layer to handle collision situations and possible data corruption.

Communication between sensor nodes and the master was tested indoors for finding out the average operation range. At a range of one to two rooms (about 15 to 20 meters) the communication did not drop any packets. At larger distances (about 30 to 40 meters) with a few walls between the sensor node and the master node, some packages were dropped, but the communication still worked at an acceptable level. When the distance was about 50 meters and there were several walls no packets were received. The most significant factor of operating range was the thickness and amount of walls. In the worst case, no packets were received when the sensor was on another floor and the distance was less than 30 meters. Our test environment was an old factory building converted to office use. The walls and floors are somewhat thicker than in an average building, which could affect the results.

We also ran stress tests against the master node's web service interface. The concurrent communications between master node and sensor nodes had only a minor effect on the overall performance of the web service. Based on our observations the Galileo board reserved about half of the CPU time for the Arduino process that performed the communication between the nodes. For use cases with low amount of clients, the limited CPU capabilities were not a problem. With a larger amount of concurrent clients (more than 50) the service experienced a noticeable increase in response times. Even higher amount of clients (more than 150) the latency became excessive and the server occasionally dropped the client connections.

Each of our sensor nodes contains three types of sensors: an air pressure sensor [16]; a photoconductive cell luminosity sensor (NSL-19M51) [17]; and a combined humidity and temperature sensor (AM2302) [18]. There are plenty of community created libraries and drivers for interacting with the attached hardware on Arduino, though in our case, we had some difficulties in finding useful code examples of how to utilize the sensor components we had chosen.

For the visualization of the data we utilized the methods presented in [19]. Importantly, the visualization is performed by the client's web browser displaying the JavaScript-based web page hosted on the master node. This approach is useful as it conserves the limited resources of the master node by off-loading the visualization work-load to the client side.

V. DISCUSSION

The model presented in this paper does not define the requirements for data encryption. The reason is that whether encryption is required or not largely depends on the use case. In our case, we chose ZigBee for data transmission between the master and the sensor nodes. ZigBee provides encryption by default, which is strong enough for our case. In fact, in our example use case, no encryption would be required simply because the data is not sensitive - for example, temperature and humidity can be measured by anyone, simply by entering the sensor location (room). If the sensor nodes autonomously send the details to the master node, and the master does not actively control the sensors, the requirements for encryption are lower than in the case where the master node actually controls the slaves. If any control data – orders on if, what, and how often – are sent by the master node, more thought should be put into the encryption, as well as into the authentication methods to reduce the risk of malicious use of the nodes. Often the chosen transmission method (e.g. ZigBee, Bluetooth and WLAN) can offer hardware based authentication and encryption options fit for most cases.

Another equally important issue not discussed in this paper is energy efficiency. The components chosen for the example system have low power consumption, but we have not performed extensive measurements for the power usage of the implemented system. Similarly to encryption, the model defines the requirements. In practice, the master node can usually be placed in a location, which has access to a constant power source, but the remote nodes may need to be run on battery power. For this reason one should carefully choose which components to use in the remote sensors to minimize the power consumption. In addition to component choices, the energy efficiency can be improved by the device software. The measurements performed by the sensors do not necessarily use much power, and most of the energy is spent on the wireless transmission of data. In our case, we control the power consumption by limiting the frequency and number of data transfers. In this case, there is no need to gather the measurements strictly in real-time. This makes it possible either to take measurements at a more relaxed pace (for example, every few minutes) or take measurements more often, but send the results infrequently in larger result sets. If control data is sent between the master and the sensors, the master can also control the rate of transmission. Also, if the data is preprocessed on the sensor nodes, the nodes themselves can make simple decisions on the frequency of communication. For example, if the nodes detect that there is a larger change in the measured values, there might be a need for more frequent transmission of data, but if the values stay the same, the data can be sent less frequently. In principle, there is no need to transfer any data as long as there is no change in the measurements, although in practice, status checks between the master and the sensor nodes should be performed to ensure that the sensors are alive and well.

One possible future research topic this paper does not analyze in depth is the matter of scalability. From the model's point of view, there can be an unlimited amount of sensor nodes for each master node, but often the chosen technologies pose limitations on the actual number of devices. In our tests the networks have been relatively small with a single master and only a few (less than ten) sensor nodes. With these kinds of small configurations the model has been proven to work, but it would be interesting to see what problems would arise in larger networks. Of course, reaching the technical device limits for the communication methods for testing purposes may be difficult simply because the required number of devices can be so high that it would cost too much to ever acquire enough sensor nodes. In fact, it is possible that the master node's capacity to process and store the received data would run out sooner simply because of the limited computing performance of the master's hardware.

VI. SUMMARY

This paper introduced a model for sensor networks used for gathering and distributing sensor measurements. The model consists of several sensor nodes and of a master node. The sensor nodes collect the raw sensor data, and the master node gathers the data from each of the sensor nodes, and provides the data for clients in a structured format. A proof-of-concept implementation based on the model was also introduced.

REFERENCES

[1] M. Saari, P. Sillberg, P. Rantanen, J. Soini, and H. Fukai, "Data collector service - practical approach with embedded linux," in 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, pp. 1037–1041.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” Comput. Networks, vol. 38, no. 4, Mar. 2002, pp. 393–422.

[3] A. Adriansyah and A. W. Dani, “Design of Small Smart Home system based on Arduino,” in 2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS), 2014, pp. 121–125.

[4] X. Jin, S. Chun, J. Jung, and K.-H. Lee, “IoT Service Selection Based on Physical Service Model and Absolute Dominance Relationship,” 2014 IEEE 7th Int. Conf. Serv. Comput. Appl., 2014, pp. 65–72.

[5] T. Laukkarinen, Abstracting Application Development for Resource Constrained Wireless Sensor Networks. Tampere, 2015.

[6] C. P. Kruger, A. M. Abu-Mahfouz, and G. P. Hancke, “Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components,” Proc. IEEE Int. Conf. Ind. Technol., vol. 2015-June, no. June, 2015, pp. 1926–1931.

[7] Lukas, W. A. Tanumihardja, and E. Gunawan, “On the application of IoT: Monitoring of troughs water level using WSN,” in 2015 IEEE Conference on Wireless Sensors (ICWiSe), 2015, pp. 58–62.

[8] J. Dos Santos, C. Hennebert, and C. Lauradoux, “Preserving privacy in secured ZigBee wireless sensor networks,” in 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 715–720.

[9] Zigbee Alliance, “ZigBee Specification, Document 053474r20,” 2012, Retrieved February 19, 2016 from http://www.zigbee.org/download/standards-zigbee-specification/

[10] Intel Corporation, “Intel Galileo Gen 2 Development Board,” 2014. Retrieved February 19, 2016 from http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-g2-datasheet.html

[11] Node.js Foundation, “Node.js v5.6.0 Documentation.” Retrieved February 19, 2016 from https://nodejs.org/api/

[12] Yocto Project and Linux Foundation, “Yocto Project.” Retrieved February 19, 2016 from https://www.yoctoproject.org/

[13] M. Banzi and Shiloh, “Getting Started with Arduino,” 3rd ed., vol. 11. Maker Media, Inc, 2014.

[14] MaxStream, “XBee TM Series 2 OEM RF Modules,”. 2007, pp. 1–60.

[15] M. Kooijman, Building Wireless Sensor Networks Using Arduino. Packt Publishing Limited, 2015.

[16] Freescale Semiconductor, “Miniature I2C Digital Barometer MPL115A2,” 2013. Retrieved February 19, 2016 from http://cache.freescale.com/files/sensors/doc/data_sheet/MPL115A2.pdf.

[17] L. Curvan, “Data Sheet, NSL-19M51, TO-18 Open Plastic Encapsulated,” 2008. Retrieved February 19, 2016 from http://docs-europe.electrocomponents.com/webdocs/002e/0900766b8002e0d5.pdf.

[18] Aosong Electronics Co Ltd, “Digital-output relative humidity & temperature sensor/module DHT22/AM2302,” 2015. Retrieved February 19, 2016 from https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[19] J. Soini, P. Sillberg and P. Rantanen, “Prototype System for Improving Manually Collected Data Quality,” 2014 Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2014, September 19-22, 2014, pp. 99-106.

PUBLICATION

III

Interpretation, Modeling and Visualization of Crowdsourced Road Condition Data

Sillberg, P., Saari, M., Grönman, J., Rantanen, P. and Kuusisto, M.

Intelligent Systems: Theory, Research and Innovation in Applications. Ed. by Jardim-Goncalves, R., Sgurev, V., Jotsov, V. and Kacprzyk, J. 2020, 99–119. (Extension of peer reviewed conference publication Sillberg et al., 2018)
DOI: 10.1007/978-3-030-38704-4_5

Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data

Pekka Sillberg, Mika Saari, Jere Grönman, Petri Rantanen, and Markku Kuusisto

Tampere University, Faculty of Information Technology and Communication Sciences, Pori, Finland

Abstract Nowadays almost everyone has a mobile phone and even the most basic smartphones often come embedded with a variety of sensors. These sensors, in combination with a large user base, offer huge potential in the realization of crowdsourcing applications. The crowdsourcing aspect is of interest especially in situations where users' everyday actions can generate data usable in more complex scenarios. The research goal in this paper is to introduce a combination of models for data gathering and analysis of the gathered data, enabling effective data processing of large data sets. Both models are applied and tested in the developed prototype system. In addition, the paper presents the test setup and results of the study, including a description of the web user interface used to illustrate road condition data. The data were collected by a group of users driving on roads in western Finland. Finally, it provides a discussion on the challenges faced in the implementation of the prototype system and a look at the problems related to the analysis of the collected data. In general, the collected data were discovered to be more useful in the assessment of the overall condition of roads, and less useful for finding specific problematic spots on roads, such as potholes.

1 Introduction

It is important to keep road networks in good condition. These days, technology and mobile devices in particular enable the automation of environmental observation [1, 2]. Mobile phones can be deployed for a particular purpose for which they were not originally designed. In addition, applications that combine road maintenance and mobile devices have already been developed [3]. In Finland, there has been a similar study on how to utilize mobile phones for collecting road condition information [4]. In the study, bus companies tested mobile phone soft-ware that sends real-time weather condition data to road maintainers in winter time. Nevertheless, traditional road condition monitoring requires manual effort – driving on

the roads and checking their condition, observing traffic cameras, and investigating reports and complaints received from road users. Automation of the monitoring process, for example by utilizing crowdsourcing, could provide a more cost-efficient solution.

Data gathering is an important part of research related to the Internet of Things (IoT) [5]. In this research, the focus of data gathering has been redirected toward a Wireless Sensor Network (WSN) [6] type of solution. Previously, we have studied technologies related to applications that automate environmental observations utilizing mobile devices. In a recent research study [7], we introduced two cases: the tracking and photographing of bus stops, and the tracking and photo-graphing of recycling areas. The first case used mobile phones and the second used a Raspberry Pi embedded system. Our other study [8] facilitated the utilization of information gathered from road users. As part of the research work, a mobile application was developed for gathering crowdsourced data.

The gathered data per se are not very usable and therefore some kind of processing is necessary. Ma et al. discussed IoT data management in their paper [9] and focused on handling data in different layers of WSN. Also, they discussed data handling challenges, approaches, and opportunities. In this study we use our previously introduced Faucet-Sink-Drain model [10]. In this model the data processing and data sources are combined in a controlled and systematic way.

This paper is an extension of Sillberg et al. [11], where the focus was on introducing the prototype system. In this extension paper, more emphasis is placed on the models behind the prototype system. We have developed a mobile application for sensing road surface anomalies (called ShockApplication). The purpose of this application is to sense the vibration of a mobile phone installed in a car. The application was tested by gathering data on real-life scenarios. The data were stored in a cloud service. In addition, we present methods that utilize the free map services available on the Internet for visualization of the data.

The research goal in this paper is to combine models of 1) data gathering and 2) analysis of the gathered data that enables effective data processing of large data sets. Both models were applied and tested in the developed prototype system. Our previous studies related to the models are presented in Section 3, where the data gathering model and the modifications made for this study are introduced in subsection 3.1. Data processing produces useful information for the user. Subsection 3.2 describes the processing model used in the prototype system. This model is designed as a general-purpose tool for systematic control and analysis of big data. With the use of these fundamentally simple models it is possible to create practical and interoperable applications.

The rest of this paper is structured as follows. In Section 2, we introduce the related research on crowdsourcing efforts in the collection of road condition data. Section 4 integrates the models presented in Section 3. In Section 5, we present the test setup and results. Section 6 includes a discussion and suggestions for future research on the topic and finally, the study is summarized in Section 7.

2 Background

Nowadays almost everyone has a mobile phone and even the most basic smartphones often come embedded with a variety of sensors [2]. This opens up the possibility of crowdsourcing through the use of mobile phones. The term crowdsourcing was defined by [12] in 2006. When several users use their devices for gathering data for a specific purpose, it can be considered a crowdsourcing activity. The idea of utilizing crowdsourcing as a model for problem solving was introduced in [13]. Furthermore, crowdsourcing can be used to support software engineering activities (e.g., software development). This matter has been widely dealt with in survey [14].

There have been several studies on using a mobile phone to detect road sur-face anomalies. One piece of research [15] presented an extensive collection of related studies. Further, the research introduced an algorithm for detecting road anomalies by using an accelerometer and a Global Positioning System (GPS) integrated into a mobile phone. The application was described as easy-to-use and developed for crowdsourcing, but the crowdsourcing aspects were not elaborated. The tests were performed with six different cars at slow speeds (20 km/h and 40 km/h). The route used in the test was set up within a campus area. The research paper did not discuss the visualization aspect nor the application itself and focused primarily on the algorithm that was presented.

The research presented in [16] and [17] was aimed at finding particular holes in a certain road. [16] used a gyroscope instead of an accelerometer and looked for spikes in the data. The other information logged was sampling time, speed, and GPS locations. The test was conducted on a route that was about four kilometers long and the test was repeated five times to ensure consistency and repeatability. The crowdsourcing aspect was not mentioned and, according to the paper, the data were collected “through a common repository.” The research [17] presented an Android application for detecting potholes, but did not provide much detail on the technical implementation.

There are several studies where the research was performed in a real-life scenario using taxis [18, 19] or buses [20]. In study [18], the data were gathered by seven taxis in the Boston area. The data collection devices were embedded computers running on a Linux-based operating system. In study [19], the data were gathered by 100 taxis in the Shenzhen urban region. The devices consisted of a microcontroller (MCU), a GPS module, a three-axis accelerometer, and a GSM module. The devices were mounted inside the cars and sent the data to servers over a wireless connection. The main idea of the research [18] was to collect data and then train a detector based on the peak X and Z accelerations and instantaneous velocity of the vehicle. The result reported in the paper was that over 90% of the potholes reported by the system were real potholes or other road anomalies. The crowdsourcing aspect was not mentioned, and the visualization was limited to showing a set of detections on a map. In study [20], the data were gathered by phones installed in buses. The data were projected on a map, but the amount of da-

ta collected (100 MB/week) and how this would affect a larger crowd were not discussed.

3 Two-phased Model of Data Processing

The research goal in this paper is a combination of models for 1) data gathering and 2) analysis of the gathered data which enables effective data processing of large data sets. Both models were applied and tested in the developed prototype system. With the use of these fundamentally simple models, it is possible to create highly practical and interoperable applications that can improve the overall quality of software.

The data gathering model and the modifications made for this study are introduced in subsection 3.1. The model is one type of Wireless Sensor Network (WSN) solution. In addition, the usage of the model in our previous research is introduced.

Subsection 3.2 describes the processing model used in the prototype system. The processing model is designed as a general-purpose tool for systematic control and analysis of big data. However, the model is very flexible and should fit a wide range of applications.

3.1 Data Gathering

Data gathering is an important part of research on the Internet of Things (IoT). In this research, the focus of data gathering has been redirected toward the WSN type of solution. Because we use mobile phones as sensor nodes, it could be categorized as a mobile sensor network. The advantages of a mobile sensor network have been discussed by Dyo [21]. In addition, Leppänen et al. [22] discuss using mobile phones as sensor nodes in data collection and data processing. A survey conducted in 2002 compiled the basic features of sensor networks [23].

In this study, we used the previously presented data gathering model. This model was introduced by Saari et al. [24] and it has three main parts: sensor node, master node, and cloud. The sensor node sends data to the master node. The master node collects and saves data, but does not process the data significantly. The master node sends data to the cloud service which stores the data. The data gathering model includes the following WSN features presented in [23]:

- Sensor nodes can be used for continuous sensing - When using a mobile phone as a sensor node, this is enabled by dedicated software.
- The mobile phone includes the basic components of a sensor node: sensing unit, processing unit, transceiver unit, and power unit.

- A sensor network is composed of a large number of sensor nodes - The prototype design presented in this study does not limit the number of mobile phones used.
- The network - Mobile phones have the communication network provided by telecommunications companies.

The model has been tested with an off-the-shelf credit card sized computer and other instruments [24-26]. The data collector service [25] used a BeagleBone Black computer and sensors. The embedded Linux controlled sensor net-work [24] used Arduino boards and sensors for the sensor nodes and an Intel Galileo Computer for the master node. Communication between sensor nodes and master nodes was handled with ZigBee expansion boards. The third study [26] used the model to test a low-energy algorithm for sensor data transmission from sensor nodes to master node.

Fig. 1 shows the modified data gathering model. The present study differs from previous research in that we used mobile phones for data gathering, which caused changes to the data gathering model. Another difference from the previous model [24] is that the sensor nodes and master nodes are combined into one entity. This was due to the use of mobile phones as sensor devices. The mobile phone includes the necessary sensors, data storage, and communication channels for this prototype system. In addition, the mobile phones use the Android operating system (OS), which has enough capabilities to gather and store data. Also, the communication protocols are supported by OS. We developed the testing software during this research. This software, called the ShockApplication, and its properties are described later in Section 5.1.

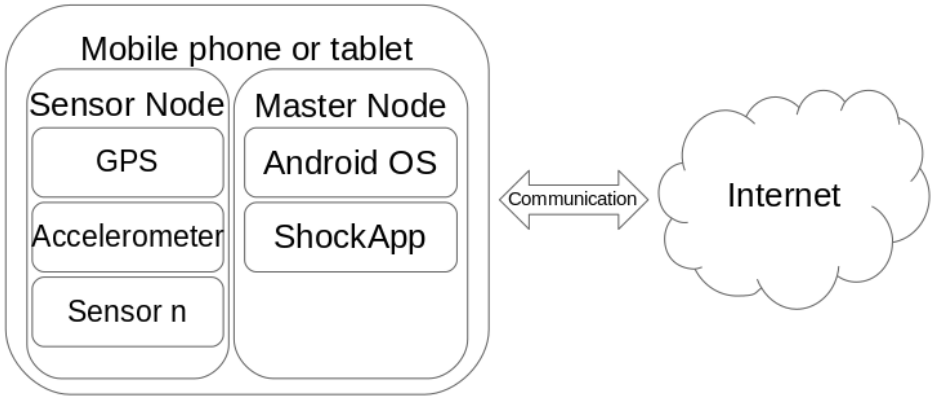

Fig. 1. The modified data gathering model.

The usage of mobile phones enabled the crowdsourcing idea. The developed ShockApplication can be installed on all modern Android phones. The user has an identification mark which helps to order the data points in the cloud. The data are stored in a cloud service.

3.2 Data Processing: Manageable Data Sources

For the data processing part, the Faucet-Sink-Drain model introduced in [10] is applied to the system architecture. The ultimate goal of the model is to enable realization of a framework that is able to manage data and data sources in a controlled and systematic way [10]. In this study, the model was applied to the prototype system, but the implementation of the framework was not carried out. This prototype is the first instance of the model in a real-world use case and will help in the further evaluation and development of the model.
The model considers that data processing can be modeled with a water piping apparatus consisting of five components: faucets, streams, sink, sieves, and drains [10]. The data flow through the model as many times as is deemed necessary to achieve the desired information. At each new cycle, a new set of faucets, sieves, and drains are created, which generate new streams to be stored in the sink. [10]

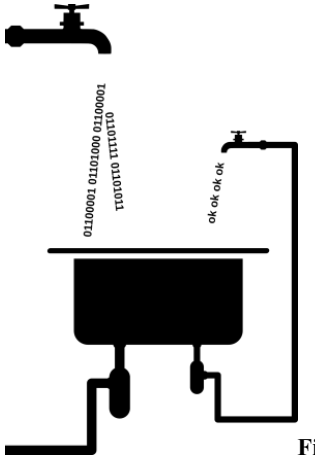

Fig. 2. Abstract data processing model. [10]

The components of the Faucet-Sink-Drain model are shown in Fig. 2. The faucet is the source of the data (e.g., original source or processed source). The running water (i.e., strings of numbers and characters) are instances of data streams, and the sink is used for storing of the data. The sieve is a filter component with the capability of selecting and processing any chunk of any given data stream. The drain is a piping system to transfer data to other locations. The drain may also be utilized for removal of excess data. [10]

The Faucet-Sink-Drain model, by design, does not specify how the data are gathered into it. As shown in Fig. 2, the initial data simply appear in the model by means of the attached faucet (or faucets). The gap can be filled by utilizing models that are stronger in this respect, such as the data collection model described in subsection 3.1.

4 Integration of the Models in the Prototype System

The models used lay out the basis for measurement and data analysis. By following them, it is then possible to implement the artifacts of the prototype system. The implemented prototype system has five identifiable high level tasks:

1. Acquisition: The data are gathered by a mobile device, which acts as a combined sensor-master node as it is capable enough for both of those tasks.
2. Storage: The cloud service receives and parses the data (communicated by the master node). Parsing of the data is the first task to be done on the system before the received data can be fully utilized. After parsing is finished, the service can then proceed by storing and/or by further processing the data.
3. Identification and Filtering: The data will be identified and filtered when the service receives an HTTP GET query on its REST (Representational State Transfer) interface. The selection is based on the rules that are passed in the request as parameters.
4. Processing: The selected data are processed further by the rules given out by the program.
5. Visualization: The data provided by the service are finally visualized in a client's user interface, e.g., web browser.

The data gathering is performed by a mobile phone by utilizing several of its available sensors. Secondly, the collected data are communicated to the cloud service where storage, selection, and further processing of the data are implemented. Once the data have been processed the last time, they are ready to be presented to the user, for example, to be visualized in a web browser or provided to another service through a machine-to-machine (M2M) interface.

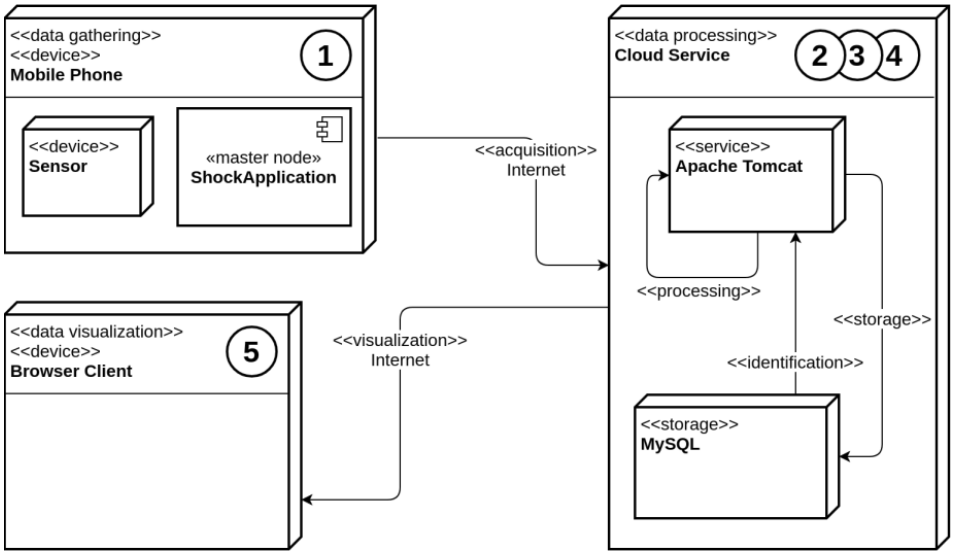

Fig. 3. System deployment diagram.

Fig. 3 shows the deployment diagram of the implemented system. It also depicts where the aforementioned tasks are carried out. These tasks can also be identified from the incorporated models, the Data Gathering model and the Faucet-Sink-Drain model. The first task, data acquisition, corresponds to the whole data gathering model and also to the combination of the (leftmost) faucet and stream icons in Fig. 2. The storage task matches the sink icon in Fig. 2. The (right-most) sieve in Fig. 2 represents the third task, identification and filtering whereas the combination of (rightmost) drain and faucet represent the processing task. The final step, visualization, is said to be handled by the sink as it is "used to store and display data" [10]. However, the visualization step could begin as early as when a data stream has emerged from a faucet and could last until the moment the data have finally been drained out from the sink.

5 Testing

The high-level description of our testing setup is illustrated in Fig. 4. The purpose was to gather data from mobile devices – primarily smartphones – that could be used to detect the surface condition of the road being driven on. These data could be further refined into more specific data, such as reports of bumps on the road, uneven road surfaces, roadworks, and so on. The traffic signs visualize the possible roadside conditions that users might be interested in. The data are sent to a central service and can be later browsed using a user interface running in a web browser.

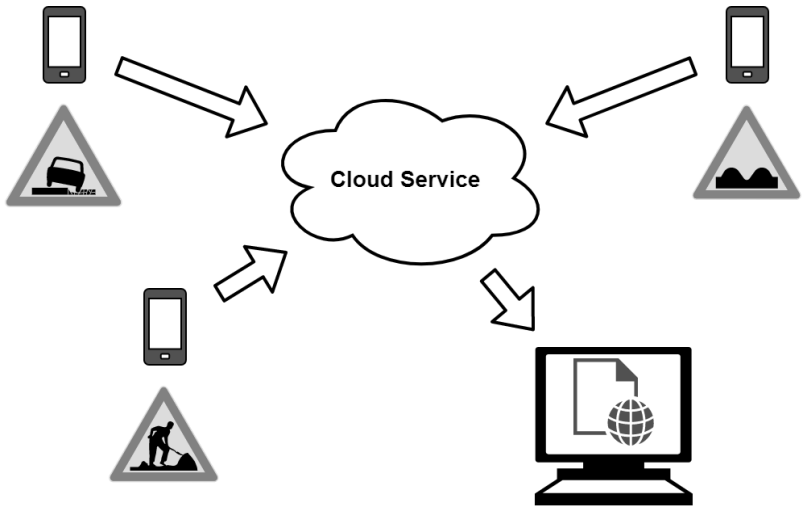

Fig. 4. High-level diagram of the test setup.

In our case, the users travelled by car. In principle, other road users such as cyclists or motorcyclists could be included, but in the scope of this study, only passenger car users were considered.

5.1 Setup

Existing studies often assume that the device is firmly attached in a specific place inside the vehicle, and in a specific way, but for crowdsourcing purposes this is not a feasible scenario. It should be possible to attach the device in a way that is the most convenient for the user, and in an optimal scenario the device could also be kept, for example, inside the pockets of the user. In our benchmarks, the device holder was not limited although we presumed that the devices were placed in a fairly stable location, and did not move about the vehicle in an unpredictable fashion (e.g., sliding along the dashboard).

In addition to the attachment of the device, several other factors (e.g., suspension, tires, vehicle load, and weight) may affect the sensor reading. It can be challenging to implement measurement of these factors in crowdsourcing scenarios. Due to these limitations, we decided to focus on sensors available in commonly used mobile devices.

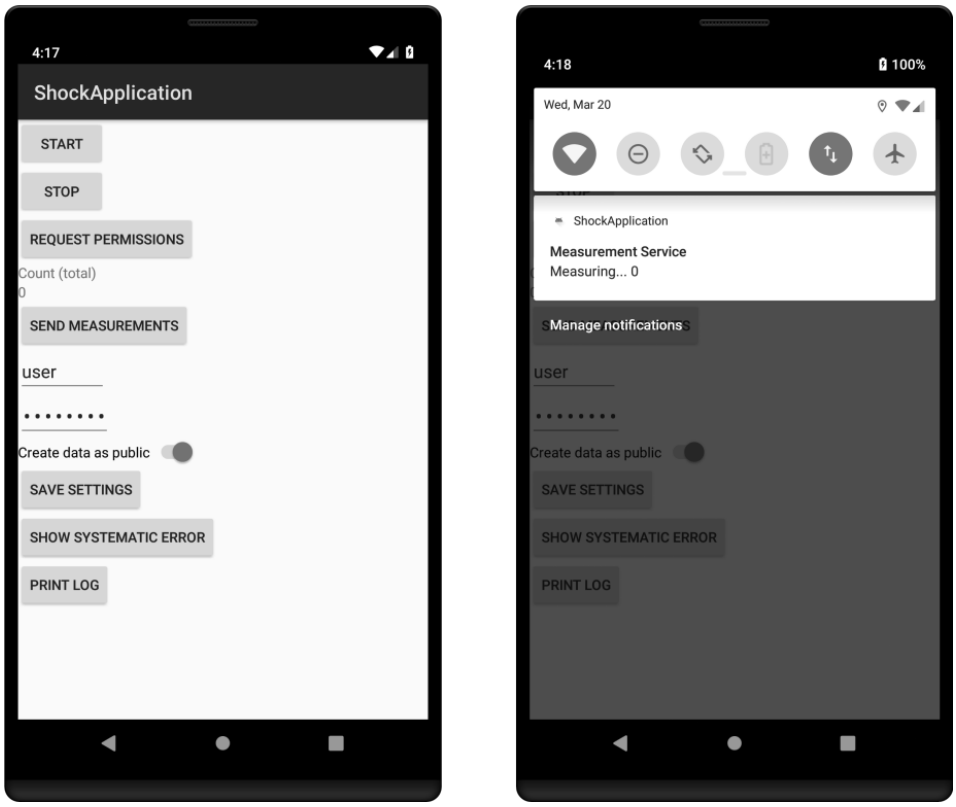

Fig. 5. The Android test client.

The testing software itself was a simple Android application, usable on any reasonably recent Android phone. Most of the newer smartphones generally contain all the necessary sensors required in our use case. The application consists of a single main view, shown on the left side of Fig. 5. In our case, the user only needs

to input his/her credentials (in the example, “user”) and use the start and stop buttons to control when the sensors are active. The user interface also contains a few convenience functions: the possibility to attempt manual transmission of all collected data; a count, which shows the total number of measurements (a single measurement contains all sensor data collected at a particular point in time, in the example pictures taken from an Android emulator the value is shown simply as “0”); the option to create all measurements as “public”, which means that any logged-in user can see the travelled route and the collected measurements; the option to save the updated settings, mainly authentication details; and two debug options that the users do not generally need to use. The software will automatically select between the linear accelerometer (which is used, if available) and the basic accelerometer. If the device is set on a stable surface the linear accelerometer should show zero for all axes and the accelerometer should show gravity, but in practice the devices showed slight variances from the expected values. The “show systematic error” option can be used to show the currently measured values and to select whether the systematic error should be removed from the values before sending the results to the service. The “print log” can be used to show a debug log of the events (such as errors) detected since application startup. It would have also been a minor matter to simply hide the debug options from the user interface, but as the primary purpose of the application was to collect data and this version of the application would not be made available for public down-load and installation (e.g., in an application store), there was no specific need to polish the user interface. Thus, the users were simply instructed to input their credentials and use the start and stop buttons, and to ignore the other options.

The sensor measurements are collected by an Android foreground service, which runs as a background process. After the service has been started, the main application can be freely closed and the statistics of the collected data (number of measurements) can be seen in the Android’s pull-down menu, which is visible on the right side of Fig. 5. In the trial, the users kept the sensors on while driving (i.e., when “participating” in the trial) and off at other times. In addition to changing the user credentials, no further configuration was required by the users.

The application was used to measure accelerometer data (X, Y, and Z acceleration), direction, speed, location (GPS coordinates), and timestamps. The collected information was automatically sent to the service at pre-defined intervals (every 30 minutes) by the background process. In addition, gyroscope and rotation data were stored on-device in an SQLite database for possible future debugging or testing purposes (e.g., for detecting braking or acceleration events, or the orientation of the device in general), but these data were not synchronized with the service.

For practical reasons (e.g., limitations in the available server capacity), the user trial was not open to an unlimited number of users. A total of ten users participated in the trial, of which half were university personnel and the other half volunteers from the staff of the City of Pori and from a company participating in our research project. The users either used their own smartphones or borrowed one from the university. The user’s choice of car was not limited, but as the users generally

drove their own cars, the selection of cars driven turned out to consist of smaller personal cars. A couple of users reported driving two different cars, so the number of cars was slightly higher than the number of users. The routes driven were a mixed set of commuting, work-related trips, and leisure. The majority of the driving involved consisted of driving from home to work, as reported by the users. This can also be seen in the collected data, as the same (identical) routes were driven on a daily basis.

Most of the driving was concentrated around the cities of Pori and Rauma, located on the west coast of Finland. Additional driving was done around the city of Tampere, which is located further inland, including the highway connecting Pori to Tampere. The distances were approximately 110 kilometers between Pori and Tampere and 50 kilometers between Pori and Rauma. Pori and Rauma are slightly smaller cities (with populations of about 85 000 and 40 000, respectively) whereas Tampere is the third largest city in Finland (with a population of about 232 000), although in the case of Tampere the routes driven were located mostly outside the city center. The routes are also illustrated in Fig. 6 (Section 5.3). The total duration of the testing period was about three months (from March 2018 to June 2018).

5.2 Results

The number of data points can be seen in Table 1, where the count and percentage figures of the data are grouped by different Shock Levels. The shock levels are arbitrary levels used for breaking down the data from the accelerometer readings. The first row ($L_{N/A}$) indicates the data points where the test device did not calculate the shock level. The highest level (L_4) represents the most intense values reported by the accelerometer. The levels can be recalculated afterwards for each device if needed. The shock levels are further discussed in Section 5.3.

Table 1. Breakdown of shock data points.

	$v \geq 0$ m/s		$v \geq 1$ m/s	
Shock Level	n	%	n	%
$L_{N/A}$	334730	69.3	312334	68.3
L_0	98367	20.4	98320	21.5
L_1	45083	9.34	42101	9.20
L_2	3419	0.71	3413	0.75
L_3	904	0.19	904	0.20
L_4	368	0.08	368	0.08
Total Count	482871	100	457440	100
Total Count with Level	148141	30.7	145106	31.7

The data point count on the left side of Table 1 includes all data regardless of the speed, and the right side omits speeds below 1 m/s. We have arbitrarily chosen 1 m/s to be the lowest speed recorded and taken into account in our test. This prevents the device from collecting data when the vehicle ought to be stationary, and helps to reduce the amount of unnecessary data.

In the further analysis of the data, only the pre-calculated shock level data where the speed is at least 1 meter per second are included (n_{LEVEL} = 145106). This represents approximately 30 percent of the total data collected. No further data have been eliminated from this data set. The relative percentage figures for each level in n_{LEVEL} are L_0 = 67.7, L_1 = 29.0, L_2 = 2.35, L_3 = 0.62, and L_4 = 0.25.

Tables 2 and 3 illustrate how the speed affects the measured shock intensity in the collected data. Rows 1 to 5 display the data of each individual level, while the last row ($L_{0—4}$) indicates the summarized information including each level. Table 2 indicates the average speed (v_{AVG}) and the standard deviation (v_{STD}) in each group. The average speed is quite similar on each level, while the standard deviation is only slightly lower on levels L_0 and L_1 than on the others. Additionally, the average speed and standard deviation of all data points (i.e., data with and without shock levels) was 68.0 km/h and 23.4 km/h. The respective values for data points without a shock level were 69.2 km/h and 21.6 km/h. The average speed and standard deviation information alone seem to support the fact that the reported shock levels occur around a speed of 65 km/h. However, when the data are further divided into speed-based intervals, the average speeds can be seen to be slightly higher, and about two-fifths of the data points are located above the 80 km/h limit.

Based on the data, it can be observed that algorithms used for detecting vibrations and road condition anomalies should cover at least the common urban area speed limits (from 40 km/h to 60 km/h) and preferably up to highway speeds (from 80 km/h to 100 km/h). In the area around the city of Pori, lower speeds were less represented than higher speeds. Thus, algorithms developed only for slower speeds would not be feasible for practical implementations.

Table 2. Average speed per shock level.

Shock Level	Speed (km/h)	
	v_{AVG}	v_{STD}
L_0	63.7	27.2
L_1	70.5	24.4
L_2	64.3	30.2
L_3	59.6	32.4
L_4	55.0	32.3
$L_{0—4}$	65.6	26.8

Table 3 displays the distribution of data points belonging to a given speed interval. There are six right-open intervals starting from 3.6 km/h (i.e., 1 m/s), and

ending at 120 km/h. The last row ($L_{0—4}$) indicates the percentage share of data in each speed interval of all data points. The bulk of the data belongs to the lowest level. The lowest level (L_0) appears to be over-represented in the lowest three speed intervals (3.6—60 km/h) whereas a small amount of the percentage share seems to have shifted from the lowest level (L_0) to the next level (L_1) in the last two speed intervals (80—120 km/h).

It seems logical that higher speeds (i.e., greater energy) create more variance in the vibration detected by the sensor, but on the other hand, levels L_2, L_3, and L_4 appear slightly less often at higher speeds. It can only be speculated whether the reason is – for example – due to the better overall condition of roads with higher speed limits, or the fact that the phone/sensor is simply not able to record everything because it is not necessarily mounted in the car securely.

Table 3. Distribution of data points per shock level.

Shock Level	Data Point Distribution Based on Speed (%) Right-Open Intervals; km/h					
	[3.6, 20[	[20, 40[	[40, 60[	[60, 80[	[80, 100[	[100, 120[
L_0	76.9	70.8	78.2	68.2	60.8	63.0
L_1	17.8	25.5	19.1	29.5	35.9	32.9
L_2	3.50	2.51	1.90	1.74	2.53	2.87
L_3	1.28	0.76	0.53	0.43	0.54	0.91
L_4	0.56	0.40	0.25	0.14	0.20	0.29
$L_{0—4}$	7.83	13.6	14.8	22.1	36.7	4.99

Speeds above 120 km/h account for a negligible amount of data points (totaling 38 data points), thus the information is not shown in Table 3. Almost three-fifths (58.8 percent) of the data points are distributed between 60 and 100 kilometers per hour. The phenomena can be explained by two facts. First, the data collection was conducted mostly on longer distance journeys on the highways between major cities, corresponding to higher speed limits and a longer time spent on the road. Second, heavy traffic in the tested area is not commonly observed. More detailed information may be retrievable if the data are observed on the user/device level rather than on the global level. In future, it might also be worthwhile recalculating the data in four levels instead of five to obtain a clearer distinction between “good road condition” data and “bad road condition” data. Currently, levels L_0 and L_1 seem to overlap, and contain both data types.

5.3 Visualization

Five levels (0-4) were used for describing the detected condition of the road. The number of levels has no specific meaning, and another amount of levels could be

chosen for more coarse or fine-tuned results. The levels are dynamically calculated per device, with level L0 being the "normal" of the device and L4 being the most extreme. In the current version of our application, the calculations do not take speed into consideration, even though speed does have an effect on the intensity of the measured values (e.g., variance). An exception to this is the exclusion of very low speed values (e.g., < 1 m/s), which could be caused by the user temporarily leaving the vehicle to walk about or be erroneous values caused by GPS inaccuracies when the vehicle is not in fact moving. In any case, even with-out utilizing the velocity data, the measured levels seem to correspond fairly accurately to the overall road conditions. Still, improved analysis of speed data could perhaps be used to further increase the accuracy of the level calculations.

In our case, the levels can be calculated either from the long-term data collected on the device (or from the data stored for testing purposes on the server), or by using a smaller data set, such as the data collected within the last 30 minutes. Ultimately, we decided to use smaller data sets when calculating the levels and showing the visualization on the map. The primary purpose of this was to minimize the effects caused by the user's change of vehicle as well as the cases where the user kept his/her device in a different holder or location on different trips. The test users also reported a few times when they had accidentally dropped the device, or the device had come loose from its holder. The former cases were fairly easy to recognize based on the reported, much higher than normal, acceleration values, but the latter cases tend to be erroneously detected as road condition problems.

In any case, the calculated levels should be fairly comparable regardless of the devices used, even when the individual values reported by the accelerometers are not. Unfortunately, rare cases where a user often changes vehicles remain a problem for detection. This problem would also be present if data were to be collected from, for example, public transportation utilizing the user's mobile devices.

The level markers and their use are illustrated in Fig. 6, Fig. 7, and Fig. 8. Fig. 6 shows a map using OpenStreetMaps, whereas Fig. 7 and Fig. 8 use Google Maps. The OpenStreetMaps implementation is slightly newer, but the features of both implementations are basically the same. One exception is the Street View functionality shown in Fig. 8, which is available only when using Google Maps. Both implementations also utilize the same underlying Representational State Transfer (REST) Application Programming Interfaces (API) provided by the cloud service.

The routes driven by the users are visualized in Fig. 6. The shock levels are illustrated by five colors (green, yellow, orange, red, and black – green being the best road condition, black the worst). The areas on the map are: the cities of Pori (top left), Rauma (bottom left), and Tampere (right). The various markers are also of slightly different sizes with the green "good condition" markers being the smallest and the black "bad condition" markers being the largest. This is in order to make the "bad condition" markers easier to spot among the data, which largely consist of green markers.

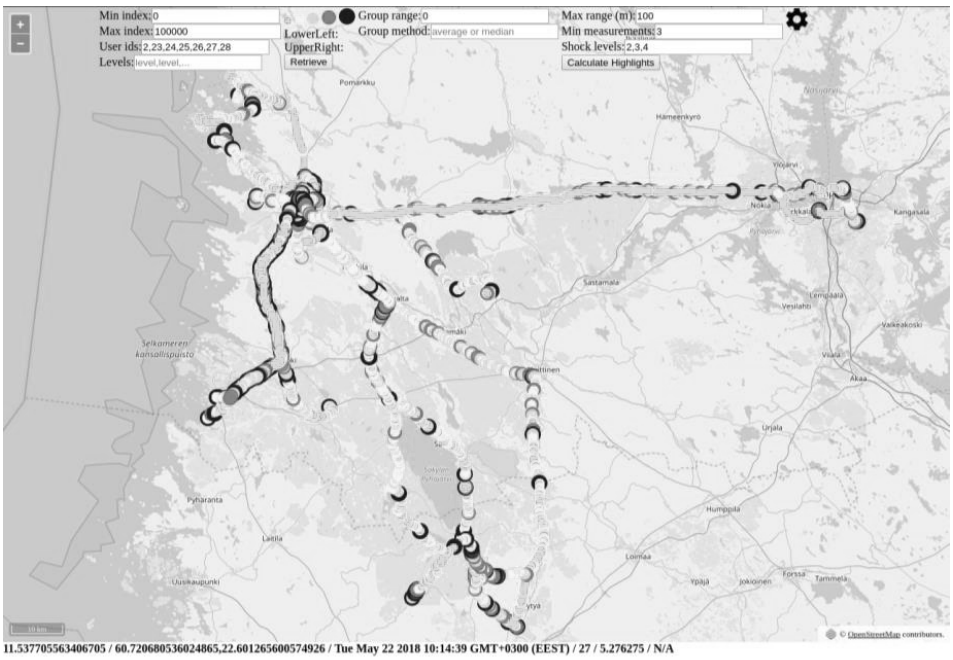

Fig. 6. Visualization of routes driven.

The user interface contains basic features for filtering data: viewing data from only a single user; excluding undesired shock levels, calculating highlights; selecting a specific date or time to observe; selecting the area to view; and the possibility to limit the number of level markers by only returning an average or median of the reported values within a certain area.

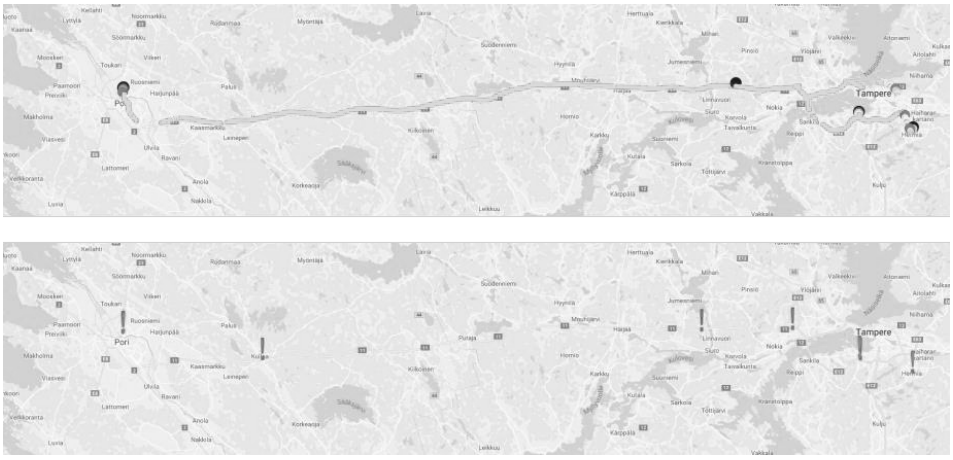

Fig. 7. Visualization of the route between the cities of Pori and Tampere.

The exclusion of undesired shock levels and highlights are illustrated in Fig. 7. The upper part of the figure shows basically the “raw data” selected from an area, in this case from a route between the cities of Pori and Tampere. In the lower part,

the individual markers are removed and only the calculated highlights (exclamation marks) can be seen. The highlights represent an area where the measurements contain a large number of certain types of shock levels. The highlights can be calculated for any level, but naturally, are more useful for spotting places where there is a high concentration of "bad condition" markers. It would also be possible to show any combination of level markers with the highlights, e.g., red or black markers without green, yellow, and orange markers.

Fig. 8. Visualization in Google Maps Street View.

Finally, Fig. 8 shows the shock level markers in the Street View application. The Street View photos are not always up-to-date so the feature cannot be used as such to validate the results, but it can be used to give a quick look at an area. In this case, the cause of several orange, red, and black – "bad condition" – markers can be seen to be the bumps located on the entrance and exit sections of a bridge located on the highway.

6 Discussion

The basic programming task of creating a simple application for tracking the user's location and gathering data from the basic sensors embedded in a mobile device is, in general, a straightforward process. Nevertheless, a practical implementation can pose both expected and unexpected challenges.

6.1 Technical Difficulties

We chose to use the Android platform because the authors had previous experience in Android programming. Unfortunately, the Android devices have hardware differences, which can affect the functionality of the application. In our case, there were two major issues. First, one of the older devices we used in our benchmarks lacked the support of a linear acceleration sensor, despite including a basic accelerometer. In practice, this means that all measured acceleration values included a gravity component without an easy or automated means of filtering the output. Filtering can be especially difficult on older models that do not contain proper rotation sensors that could be used to detect the orientation of the device.

Second, as it turned out, devices from different manufacturers and even different device models from the same manufacturer had variations in the reported accelerometer values, making direct comparison of values between devices challenging at best. Larger bumps are visible from the results regardless of the device, but smaller road surface features can become lost due to the device inaccuracies.

In practice, differences in the devices required the calculation of a "normal" for each device, against which variations in the data would be compared. Calculating a universal normal usable for all devices and users would probably be very difficult, if not entirely impossible. In any case, in laboratory conditions or in a controlled environment finding this normal is not a huge problem, but where a large crowdsourcing user and device base is concerned, finding the normal for each device can be a challenge. Additionally, the vehicle the user is driving can have a major impact on the detected values; after all, car manufacturers generally prefer to provide a smooth ride for the driver, and on the other hand, a car with poor suspension or tires can cause data variations that can be difficult to filter out. This also means that, if the user drives multiple vehicles, there should be a way for the application to either detect the vehicle used or adapt to the altered conditions.

In principle, the collected data could be analyzed to determine the device's normal, for example, if known "good condition" roads have been driven on. In practice, the data amounts (and the required server and network capacity) can be too extreme for this approach to be feasible. A better option would be to analyze the data on-device and the devices should only send the variances that exceed the calculated threshold values (i.e., detected potholes, roads of poor quality).

6.2 Interpretation of the Data

When examining the collected data set, the known places of data variance are visible, and in expected places. These include, among others, known roadworks, speed bumps, and bridge ramps, i.e., spots that the drivers cannot avoid can be easily seen in the collected data. Unfortunately, the same cannot be said about potholes or other larger, but in general, more infrequent road condition issues

which are not always detected. We did not perform extensive studies to discover the driving habits of the users participating in our trial, although a quick interview revealed (perhaps unsurprisingly) that the drivers had tried to avoid driving into potholes.

In the initial phase of data analysis, validating the findings proved troublesome. As the drivers could drive along any road they wished, we did not have a clear idea of which of the roads driven were in bad shape or where on the road the bumps were located, nor was there available any conclusive database of speed bumps or other purpose-built road features that could be accidentally identified as road surface problems. Driving to the location of each detected bump for validation purposes in the case of a larger data set would be quite impractical. To get a basic idea of where the "bumpy" roads were located, the preliminary results were shared with the department of the City of Pori responsible for road maintenance and compared with their data. The data collected by the city are based on complaints received from road users or reported by the city maintenance personnel driving on the city roads. Thus, maintaining the data requires a lot of manual labor and the data are not always up-to-date. Nevertheless, this did give us some insight into the known conditions of the roads around the city. Furthermore, the discussion with the maintenance department gave a clear indication that an automated method for the collection of road condition data around the city would be a great help for the people responsible for road maintenance.

Moreover, collecting a sufficiently large data set with a very large user base could ultimately help in finding individual road problems as drivers would, for example, accidentally drive into potholes, but in our trials identifying specific road problems turned out to be quite challenging. On the other hand, the results showed, in a more general fashion, which of the driven roads were in the worst condition, and furthermore, which parts of a single road were in worse condition than the road on average. Both findings can be used for assessing road conditions, and with a much larger data set, even individual bumps could perhaps be more reliably detected.

A larger database is also advantageous in the elimination of unwanted data caused by individual random events – such as the user moving or tapping the phone during driving, sudden braking events or accidents – which could be erroneously detected as road condition problems. On the other hand, larger sets increase computing resource requirements and challenges in managing the data. In fact, even the amount of data collected in our user trials can be problematic. One of the main challenges is the visualization of large data sets.

For testing and validation purposes, all data generated by the mobile devices were stored on our server. Storing the "good condition" data can also help to map the roads the users have driven on as opposed to only reporting detected variations from the normal. Unfortunately, serializing the data – using JavaScript Object Notation (JSON) or Extensible Markup Language (XML) – and showing the measurements on a map in a web browser may be quite resource-intensive. Even when measurements are combined and indexed on the server to reduce the amount of transferred data, there can still be thousands of markers to be drawn on the map,

especially if "good condition" data are included. Showing multiple roads in a large area simultaneously on a map can be a good method from a visualization point of view, but it can also make the web user interface sluggish or slow to load. For reference, loading and showing the map visible in Fig. 6 consisting of 100 000 measurement markers takes approximately 3—4 minutes, which is not an entirely impractical length of time for constructing the visualization, but can be an annoying delay when performing repeated work on the data set. Con-structing visualizations with smaller data sets (e.g., less than 10 000 data points), depending on the chosen filter settings, takes anything from a couple of seconds to almost half a minute.

6.3 Future Studies

One possible future action could be to open up the collected data for further analysis by other researchers. In general, the data are relatively easy to anonymize and do not contain any hard-coded user details. A method of generating anonymous data is also an advantage if a larger, more public user trial is to be performed in the future. Running the trials with a larger userbase would be one possible course of future action, although acquiring sufficient server resources for a wide-scale user trial could pose a challenge.

A less resource-intensive option could be to collect data for a longer period on a specific set of roads with the goal of discovering whether a gradual worsening of road conditions can be detected or how the results differ between winter and summer. Our current trials were run in spring and summer, and it is unknown how winter conditions would affect the results. Furthermore, the roads driven on were primarily paved and gravel roads were not included in the analysis of the data.

In addition, the increase in the number of dashboard cameras installed in vehicles, and the decrease in the prices of 360-degree cameras could provide an interesting aspect for data collection. The utilization of cameras could also make data validation easier during the trial phase, as there would be no need to go and check the detected road condition problems locally, or to use Google Street View or similar applications that may contain outdated images.

The Faucet-Sink-Drain model was used for the first time in an actual use case, and it could prove useful in other applications as well. However, the model requires more research and development to fully unlock its potential. Also, the framework [10] that is based on the model would require an actual implementation before more conclusions can be drawn of the model's usefulness.

Data security is an important factor that has not been addressed in this study. The prototype has basic user identification with username and password, but this was not used for filtering input data. Issues of data security, privacy, and anonymization of data need to be solved before commercialization.

7 Summary

This paper introduced a study that utilized data collected by sensors – primarily from an accelerometer and GPS – embedded in smartphones for detecting the condition of road surfaces. The data were obtained from a group of users driving on paved roads in western Finland. Furthermore, the test setup was described including a discussion on the challenges faced.

This paper showed how to combine a data gathering model and a data analysis model. Both of the models were applied and tested in the developed prototype system.

The results achieved from the trial period showed that even though the chosen methods could, in principle, find individual road surface problems (such as potholes), the results were more useful in the assessment of the overall condition of the road. In addition, the paper presented methods for visualizing road condition data collected from test users.

References

1. Krommyda, M., Sdongos, E., Tamascelli, S., Tsertou, A., Latsa, G., Amditis, A.: Towards Citizen-Powered Cyberworlds for Environmental Monitoring. In: 2018 International Conference on Cyberworlds (CW), pp. 454–457 (2018)
2. Satoto, K.I., Widianto, E.D., Sumardi., S.: Environmental Health Monitoring with Smartphone Application. In: 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 281–286 (2018)
3. Pyykonen, P., Laitinen, J., Viitanen, J., Eloranta, P., Korhonen, T.: IoT for Intelligent Traffic System. In: 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 175–179 (2013)
4. Yle Uutiset: Lapin Ely lupaa vähemmän lunta ja polanteita – Bussinkuljettajat keräävät tietoa Lapin teiden kunnosta. https://yle.fi/uutiset/3-9277596 (2016) Retrieved 27th. June 2018
5. Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., Jubert, I., Mazura, M., Harrison, M., Eisenhauer, M., Doody, P.: Internet of Things Strategic Research Roadmap. http://www.internet-of-things.no/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf (2009) Retrieved 23rd. March 2019
6. Hać, A.: Wireless Sensor Network Designs. Chichester, UK: John Wiley & Sons, Ltd. (2003)
7. Grönman, J., Rantanen, P., Saari, M., Sillberg, P., Jaakkola, H.: Lessons Learned from Developing Prototypes for Customer Complaint Validation. Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Serbia (August 2018)
8. Rantanen, P., Sillberg, P., Soini, J.: Towards the utilization of crowdsourcing in traffic condition reporting. 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Croatia, pp. 985–990 (May 2017)
9. Ma, M., Wang, P., Chu, C.-H.: Data Management for Internet of Things: Challenges, Approaches and Opportunities. In: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 1144–1151 (2013)

10. Sillberg, P.: Toward Manageable Data Sources. Information Modelling and Knowledge Bases XXX, Frontiers in Artificial Intelligence and Applications, vol. 312, IOS Press, pp. 101–111 (2019)
11. Sillberg, P., Grönman, J., Rantanen, P., Saari, M., Kuusisto, M.: Challenges in the Interpretation of Crowdsourced Road Condition Data. In: International Conference on Intelligent Systems (IS) (2018)
12. Howe, J.: The Rise of Crowdsourcing. https://www.wired.com/2006/06/crowds (2006) Retrieved 27th. June 2018.
13. Brabham, D.C.: Crowdsourcing as a Model for Problem Solving. Convergence: The International Journal of Research into New Media Technologies, vol. 14, no. 1, pp. 75–90 (February 2008)
14. Mao, K., Capra, L., Harman, M., Jia, Y.: A Survey of the Use of Crowdsourcing in Software Engineering. Technical Report RN/15/01, Department of Computer Science, University College London (2015)
15. Yi, C.-W., Chuang, Y.-T., Nian, C.-S.: Toward Crowdsourcing-Based Road Pavement Monitoring by Mobile Sensing Technologies. IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 4, pp. 1905–1917 (August 2015)
16. Y. A. Alqudah and B. H. Sababha, "On the analysis of road surface conditions using embedded smartphone sensors," in 2017 8th International Conference on Information and Communication Systems (ICICS), pp. 177–181, Jordan, April 2017.
17. Carrera, F., Guerin, S., Thorp, J.B.: By the People, for the People: The Crowdsourcing of "STREETBUMP": An Automatic Pothole Mapping App. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS), vol. XL-4/W1, no. 4W1, pp. 19–23 (May 2013)
18. Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The Pothole Patrol. In: Proceedings of the 6th International Conference on Mobile systems, applications, and services - MobiSys '08, Colorado, USA, p. 29 (June 2008)
19. Chen, K., Lu, M., Tan, G., Wu, J.: CRSM: Crowdsourcing Based Road Surface Monitoring. In: 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, China, pp. 2151–2158 (November 2013)
20. Alessandroni, G., Klopfenstein, L., Delpriori, S., Dromedari, M., Luchetti, G., Paolini, B., Seraghiti, A., Lattanzi, E., Freschi, V., Carini, A., Bogliolo, A.: SmartRoadSense: Collaborative Road Surface Condition Monitoring. The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), Italy (August 2014)
21. Dyo, V.: Middleware design for integration of sensor network and mobile devices. In: Proceedings of the 2nd International Doctoral Symposium on Middleware - DSM '05, New York, New York, USA: ACM Press, pp. 1–5 (2005)
22. Leppanen, T., Perttunen, M., Riekki, J., Kaipio, P: Sensor Network Architecture for Cooperative Traffic Applications. In: 2010 6th International Conference on Wireless and Mobile Communications, pp. 400–403 (2010)
23. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: a Survey. Computer Networks, vol. 38, no. 4, pp. 393–422 (March 2002)
24. Saari, M., Baharudin, A.M., Sillberg, P., Rantanen, P., Soini, J.: Embedded Linux Controlled Sensor Network. In: 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1185–1189 (2016)
25. Saari, M., Sillberg, P., Rantanen, P., Soini, J., Fukai, H.: Data Collector Service - Practical Approach with Embedded Linux. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1037–1041 (2015)

26. Baharudin, A.M., Saari, M., Sillberg, P., Rantanen, P., Soini, J., Kuroda, T.: Low-Energy Algorithm for Self-controlled Wireless Sensor Nodes. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 42–46 (2016)

PUBLICATION
IV

Lessons learned from developing prototypes for customer complaint validation

Grönman, J., Rantanen, P., Saari, M., Sillberg, P. and Jaakkola, H.

*Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*2018, 27–30

Lessons Learned from Developing Prototypes for Customer Complaint Validation

JERE GRÖNMAN, PETRI RANTANEN, MIKA SAARI, PEKKA SILLBERG AND HANNU JAAKKOLA, Tampere University of Technology

This research introduces two prototypes installed in vehicles and a cloud service for autonomous collection of data. The prototypes utilize camera, location data, and timestamps to help those responsible for managing customer complaints, and to improve the overall quality of the provided customer service. The use of the system is illustrated by two cases: tracking and photographing bus stops, and tracking and photographing recycling areas. The first prototype is implemented for the Android mobile platform and the second one for the Raspberry Pi single-board computer. This paper discusses the differences and challenges faced in designing and implementing the two prototypes for different platforms.

1. INTRODUCTION

The Internet of Things (IoT) is the expansion of Internet services, which connects everyday physical objects to a network. This connection between network and physical world objects makes it possible to access remote sensor data and to control the physical world devices from a distance. One study addressing the IoT, which is cited quite often, is "The Internet of Things: A survey" [Atzori et al. 2010].

In this research, the focus has been redirected toward the Wireless Sensor Network (WSN) type of solution. The basic features of sensor networks were compiled in a survey by Akyildiz et al. in 2002. In this research, we present two different prototypes for collecting data. These prototypes were designed as nodes of WSN. The design processes were iterative and the main goal was to improve the prototype in every iteration round. This study is a "lessons learned" type of research on software quality and prototype testing, where we present the problems encountered and the solutions to them. This research is a continuation of our research into different areas of IoT [Saari et al. 2016; Saari et al. 2017; Grönman et al. 2018]. Often, the Agile method is used more than the traditional plan-driven methods (such as the Waterfall method) when developing prototype systems. The authors of this paper have discussed the challenges of modeling in an earlier study [Jaakkola et al. 2016].

The motivation for this study came from two transportation companies. Their customers often complain that the service is not at an acceptable level (e.g. the bus was not on time, or did not stop; trash was not collected on time). For companies, it can be difficult to ascertain the validity of the complaints, possibly causing unnecessary expenses when repeated complaints occur. This study and the two use cases presented in this paper illustrate configurable conditions for area observations (based on location, speed of the vehicle, cameras, and other sensors). In the past, the drivers photographed locations and made observation reports manually, but this process turned out to be tedious and error-prone. Thus, it was decided to design a system that could work autonomously without input from the driver. The companies can use the collected data to validate customer service requests/complaints, and to improve the overall quality of the provided customer service.

Author's address: J. Grönman, Tampere University of Technology, Pori, P.O. Box 300, FI-28101 Pori, Finland; email: jere.gronman@tut.fi.

In: Z. Budimac (ed.): Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Novi Sad, Serbia, 27–30.8.2018. Also published online by CEUR Workshop Proceedings (http://ceur-ws.org, ISSN 1613-0073)

There has been a lot of research on position systems such as vehicle tracking systems. For example [Lee et al. 2014; Jisha et al. 2017] introduced vehicle tracking systems, where the location data are stored to the database or cloud and the data could be shown with an Android mobile application. Jisha et al. deal with bus tracking systems. In these studies, the focus was on real-time tracking using the Global Positioning System (GPS). The main idea was a tracking service for customers, and the quality assurance of the transportation service was not discussed.

In our use cases the sensor nodes (mobile phones and Raspberry Pi 3 computers) send the data to the cloud service. The idea and the model of the data gathering node system were introduced in [Saari et al. 2015]. The rest of this paper is structured as follows. In Section 2, we outline the research environment and its components. In Section 3, we describe the "bus stop" case and in Section 4 the "garbage truck" case. Section 5 includes a discussion where the findings of this prototype development process are handled. The study is summarized in Section 6.

2. HIGH-LEVEL ARCHITECTURE

The goal of the system is to provide a tool for those processing customer complaints. In our case, two use cases acted as pilot studies for testing the functionalities of the system. The first case, "bus stops," consists of tracking a bus traveling on the route by collecting images, location data, and timestamps. The bus company participating in our pilot study reported that common complaints reported by customers are about the bus not arriving on schedule (too early, too late or not arriving at all) or the bus not stopping even though there were people waiting at the bus stop. The latter issue especially can be difficult to validate, and the bus company was interested in improving the quality of their bus service by finding out if and when the complaints reported a real problem.

The second use case, "garbage truck," collected the same data (images, location, timestamps). The purpose was to keep track of when the garbage truck visited the recycling area and if the bins were not emptied, and whether there was something in the area that prevented the truck from doing its work. In some cases, pictures were already being taken by the garbage truck drivers in the area around Pori, but this is, in general, manual work, and capturing and managing the pictures can be tedious and error-prone. Similarly to the bus company, the company running the garbage collection receives complaints about the quality of the work, and the company was interested in an automatic system for collecting data around the recycling sites to validate the complaints.

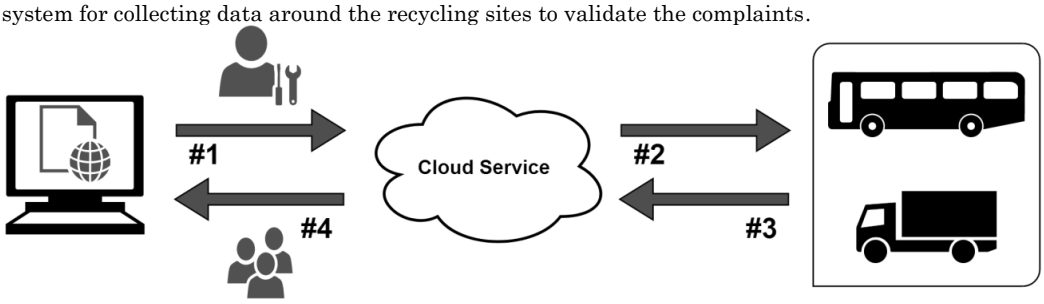

Fig. 1. High-level diagram of the system.

Figure 1 illustrates the high-level architecture. The system consists of a central service, which provides representational state transfer (REST) application programming interface (API) for client-side interaction and remote procedure call (RPC) functionality for delivering tasks to (back end) devices or for submitting task results. A simple web portal is implemented using Hypertext Markup Language (HTML) and JavaScript, which interacts with client-side methods. The web user interface

allows the user to create tasks (#1, Figure 1), which contain the pre-conditions for device operation, the selection of devices that participate in the tasks, and the desired output parameters. The tasks are delivered to the target devices (#2, Figure 1).

The pre-conditions contain options such as coordinates (or areas), time intervals (e.g., only take pictures during working hours), and velocities (e.g., should the device be moving at a certain speed or stationary). The output parameters notify the devices as to what information should be returned in the result responses (#3, Figure 1), such as pictures or location data – by default, all responses must contain timestamps.

The results can be returned in near real-time or in batches. In our use cases, there is no need for immediate responses and in general, the results can be returned at a time most convenient for the device as long as the results arrive within a reasonable time period (for example, within 24 hours). In general, the tasks do not contain information on the expected amount of output data. In our use case, most of the transmitted data consists of captured images. The amount of images is highly dependent on the speed at which the vehicle is moving and how long the vehicle stays within the designated area. The prototype application will attempt to compensate for the vehicle velocity (e.g., by capturing more pictures when the device is moving faster), but the tasks themselves do not contain any guidance for this functionality. The primary reason for this is that the device is better equipped to estimate its own capabilities and features than the service and providing overly detailed parameters would only complicate the tasks by requiring individual customization for each device.

The submitted results (#3, Figure 1) are indexed in the service and can be freely browsed by the user (#4, Figure 1), for example, by selecting a bus stop (coordinate) and the time reported in the complaint. The core service uses a platform developed in a previous project [Iftikhar et al. 2018], and for historical reasons messages based on the Extensible Markup Language (XML) are used, though other formats (e.g., JavaScript Object Notation) could be used in our use case as well.

In our current implementation no further image analysis is performed either on-device or in the service. In principle, it would be beneficial if image processing could be utilized to detect problems reported in customer complaints. In practice, the variations in environments (location, time of day, snow, rain, etc.) and different use cases (detection of people, undesired objects) make it very challenging to develop a reliable algorithm. As long as a reasonable amount of pictures is shown to the person responsible for processing the complaints, a human observer can detect problems by looking at the pictures

3. USE CASE: BUS STOPS

In the first use case, a prototype was installed in a bus traveling along a bus route within the City of Pori. The prototype is fully autonomous and does not require any input from the bus driver. The stops on the route were assigned to the prototype as GPS coordinate targets. The program code reads GPS coordinates continuously and compares them to the assigned targets. When the coordinates match, a picture is taken. The program code utilizes an implementation of the Haversine Formula, which determines the great-circle distance between two locations and is relatively simple to implement yet accurate enough for our use cases. The application can be installed on any reasonably new Android device and takes advantage of the built-in sensors and camera of the device.

The prototype keeps taking pictures in a predefined interval as long as it remains within the range of the target. The prototype scales both the time interval and the range from the target based on the speed of the bus to enable taking pictures at varying speeds, and also to compensate for the delay in waking up the camera. The idea was to take pictures of the approach to the bus stop to find out whether customers were present at the bus stop and also to obtain evidence (photos, timestamps, and location data) that the bus had passed – or stopped at – the bus stop on a certain date. The approach to one bus stop is illustrated in Figure 2.

Fig. 2. Approach to a bus stop showing pictures taken at varying distances from the target.

A background process was created in the application for sending the pictures after capture, although, depending on the network connection speed, the upload may not be real-time. A route with fewer bus stops and a smaller number of customers was chosen for the first prototype. In our case, the route had a few dozen stops, but within the city limits some routes may have several hundred stops (especially if the route is traveled in both directions). Furthermore, the local bus company was instructed to provide us routes with higher than average amount of complaints.

4. USE CASE: GARBAGE TRUCK

In this use case, a prototype was installed in a garbage truck. The truck has a predefined route where there are certain recycling areas nearby shopping centers. The locations were assigned to the prototype as GPS coordinate targets. A route with frequently visited targets was provided by the garbage truck company participating in our project. The goal was to select targets of varying size (a gas station, a supermarket, and a larger shopping center) located around the City of Pori. The location tracking was performed in an identical fashion to the "bus stop" case, with the applicable code re-written in Python. Similarly, the prototype is fully autonomous and does not require any input from the driver.

The prototype consists of a combination of a Raspberry Pi 3 single-board computer and commonly available sensor components (Adafruit Ultimate GPS HAT and Raspberry Camera Module V2 NoIR). Its operating system is Raspbian Stretch and the program code was made in Python, which is one of the commonly used languages for prototyping with Raspberry Pi. The prototype requires a 3G/4G - wireless modem to establish an Internet connection via Wi-Fi.

Fig. 3. Three pictures taken from the recycling area. On the left: in daylight; in the center: at night; on the right: a blocking obstacle.

In this use case, a connection to the cloud service was established once a day. During the test period of three months, more than 6500 pictures were taken of the targets. Pictures were taken both in daylight and at night. Figure 3 presents a comparison between day and night. Figure 3 also shows

a situation where an obstacle, in this case, a car, is blocking the truck's access to the recycling bins. The first two pictures in Figure 3 present normal daily operation, but in the case of the third picture, the car could have prevented the truck from emptying the garbage bins, possibly causing later complaints from customers about full containers.

5. LESSONS LEARNED

The development process for the use case prototypes was iterative in nature. Our goals were to both validate our ideas and to ascertain in a short period of time which technical solutions would work in realizing the prototypes.

One approach would have been to install cameras at each location, but in practice this was not feasible. In both cases all locations were outdoor locations and it would have required a considerable effort to guarantee the availability of electricity, and that the devices would not get wet, vandalized, or broken in the cold weather. The assumption was that installation in-vehicle would be easier. A minor concern was the operating temperature as the vehicles would be stored outside in Finnish winter when not in use. The Android implementation was not tested in wintertime, but there were no problems with the Raspberry Pi during the three-month trial run. The device itself did not contain ambient air temperature sensors, but the average temperature during the December-February period was slightly below zero Celsius with the coldest nighttime temperature reaching -21 °C in February [Foreca 2018]. The device was always on during the trial, running on the continuous power provided by the vehicle batteries. This approach also reduced the risk of the device failing to boot up due to cold weather. In the case of the garbage truck, obtaining constant power was a simple matter of using the cigarette lighter plugs, but, in the bus, re-wiring was necessary as the connectors inside the bus did not provide electricity when the main power was turned off.

A bigger problem than electricity was the attachment of the devices (both Android and Raspberry Pi) to the vehicle. In our case we selected a garbage truck that loaded the trash using a lift located in the front of the vehicle. The company also had vehicles that were loaded from the back, but this would have meant that the truck would have approached the recycling area in reverse, requiring camera installation at the back of the vehicle – possibly on the outside of the vehicle. For simplicity, it was decided to only use a camera to take photos through the windscreen. This left the rare case when the vehicle would be approaching the location from an unusual angle, e.g., around the corner of a building, from the side or from an otherwise bad direction for taking pictures through the windscreen. Especially in the bus stop case, there were no pictures available of every bus stop, and even if there had been, we did not want to make individual setups for hundreds of locations. Thus, it was impossible to know how the vehicle would approach each location. Regardless, it was decided to choose the windscreen approach to get the testing underway.

In practice, this approach provided more problems than expected. Initially, there was slight concern about reflections on the glass surface. In practice, this turned out not to be a big problem, because the camera would take several pictures when the vehicle was approaching the location and major reflections did not occur often enough to pose a real problem. However, a more serious problem was how to install the devices in the vehicles.

The curved windscreen of the truck and bus made the traditional suction cup-based attachments unusable. The vibration and movement of the vehicle caused the device to fall off of the window. Additionally, a permanent installation of the prototype was not desirable as there might have been a need to remove the device during testing. It would also have been impractical to make extensive modifications to the vehicles because the vehicles were in regular use by the companies. The installation of the Raspberry Pi was slightly easier as the camera as well as the GPS antenna could be detached and installed in a different place to the device itself. This meant that the components that needed to be installed near the windscreen were more lightweight than a smartphone, which

contains all the components in one package. The ideas for a final prototype installation ranged from ordering various attachment holders from the Internet to using a 3D printer to create a custom casing. In the end, the installation consisted of a lot of two-sided tape. The solution was not pretty and was passable at best. All in all, our primary concern was the device itself, the software, and testing our idea, but perhaps a little more thought should have been put into how to setup the device in a real environment regardless of the trial nature of the tests.

The members of our research team had previous experience in programming applications for the Android platform and also in using the Java programming language. Furthermore, our research team had readily available Android devices - both personal devices and devices provided by the university – that could be used in prototype development and testing. This meant that the prototype development process could be started without the need to learn the basics for a new platform. The two most popular mobile platforms (Android and iOS) provide similar starting points for our use case requirements: APIs for controlling the camera, accessing the Internet, and tracking the device location, making the choice mostly about developer preference.

Creation of a simple application for tracking the location, firing the camera in pre-designated coordinates, and uploading the pictures to a remote service was relatively simple – the APIs are well documented and a simple web search provides plentiful examples for common use cases. In general, only two problems were met related to the programming.

Firstly, we used a pre-existing service developed for previous research projects, which utilized the XML-based data format. Unfortunately, by default, the Android platform does not support standard annotation-based class definitions (e.g., Java Architecture for XML Binding), which meant that we could not directly use the same Java code as we had used in previous Java applications. This is an example of one of the generally minor problems caused by the fact that Android does not provide full API compatibility with Oracle's Java. The problem was fixed by creating an XML parser using the Android's XML pull parser and serializer, which are relatively simple to use though perhaps are slightly more error-prone by requiring modifications to the parser code when the format is modified as opposed to annotation-based solutions, which only require modifications to the class declarations.

The second programming related issue was with our implementation of the camera use. For some unknown reason, especially on older Android devices (Nexus 7 tablet and Samsung A5), using the camera repeatedly and sometimes in quick fashion caused application crashes or the camera got "stuck," capturing only a black screen. Fixing the issue required several attempts with various programming solutions and the implementation was never as stable as we had hoped for.

The issues with stability caused an additional problem. In our initial trials, a member of our research team was present in the bus, and could make corrections or restart the application when problems occurred, but in the future this would not be the case. In the next phase, the device would be installed in the vehicle for a period of three months, during which there could be a need for fixing problems and to further improve the prototype software. Repeatedly visiting the company for prototype maintenance would be a tedious process for the research team and also problematic for the company as their vehicles were in use on a daily basis.

Remotely accessing the mobile device, for example, for restarting an application or installing a new application version was a real challenge. As our application was not available in the Google application store, we could not take advantage of the remote installation options provided by the store, and directly accessing the device over the public Internet – i.e. accessing the dynamically assigned Internet protocol (IP) address – would have been difficult without developing extensive support mechanisms. This was one of the primary reasons (in addition to the unstable camera implementation and problems with installing the device to the vehicle) for dropping the Android implementation and looking for alternative options.

The Raspberry Pi-based solution provided much needed help for the remote access problem. The device is, in practice, a Linux-enabled computer, which means that many of the methods available

for desktop application development are available. In our previous projects we have had some experience with Raspberry Pi in particular, making it a logical choice, though many of the other single-board computer solutions available on the market should work as well. In any case, by using a dynamic Domain Name System (DNS) update it was possible to keep track of the public IP address assigned by the Internet provider. It was also possible to directly use version control (in our case, sub-version) as a "cheap alternative" for deploying new application versions on the device, and access to the device can be achieved using Secure Shell (SSH). On Linux, the applications can also be easily set up to start up on boot or at designated intervals either as services or by utilizing crontab. Crontab was also utilized to run scripts that periodically checked whether our application was still alive, logging the application status, and restarting the application if it had crashed.

OpenJDK is readily available for Linux and can also be used with the Raspbian operating system, enabling the use of Java applications. The advantage was that most of our previously written utility code (accessing the Internet, XML parsing from our server-side implementation, etc.) could be directly used on Raspberry. The disadvantage was that all code that accessed the device sensors and the camera would be re-written as no compatible APIs existed, and the preferred programming language was different (Java vs. Python). Accessing the camera (using the raspistill command line tool) or GPS data (using the gpsd service daemon) with Python is not difficult, though the level of API documentation is not on a par with the Android documentation. It can also be more challenging to find pre-made examples. Many of the example projects found online are, for the lack of a better word, "hacks", and the re-usability of code is more difficult than on the commonly used mobile platforms. This is also one issue that one should keep in mind when deciding which platform to use for rapid prototyping. On the positive side, a more low level API access is available on Raspberry Pi, if such functionality is required.

An important note is that remote access also creates a potential security vulnerability, which should be taken into account, especially when using potentially unstable or vulnerable prototype or development versions of applications. Using a dynamic DNS service also seems to create a hot spot for attempts at breaking into the device using dictionary and brute-force attacks. As a minimal configuration, the default SSH port and passwords should be changed and remote root access disabled. In our case, we used a separate 4G modem because Raspberry Pi does not provide a 3G/4G connection, and the modem was also set up to work as a firewall.

This remote access approach worked fairly well even though there were a few minor problems. Around the heavy industry area and the power plant located in Pori there were problems with cell reception and data transfer. The modem initially chosen also had issues with energy management. Regardless of the configuration options, after a longer period of inactivity the modem would go into a power-safe state, cutting remote access to the Raspberry Pi and sometimes the modem would "hang" requiring a physical restart. Periodically pinging the remote server seemed to fix both issues, though it would have been better if the modem had been configured to function as intended. Another minor issue with the modem was that if power were lost for whatever reason, the modem would not automatically connect to the Internet, and would instead require the user to press a button on the device. In our case a continuous power supply was made available both in the bus and in the truck to fix the issue. Nevertheless, it became clear that it can be challenging to figure out without testing how well a specific modem will work in various conditions and what configuration options are available, especially if cheaper devices targeted to end-user customers are utilized.

6. CONCLUSIONS

This paper presented a high-level diagram for a service designed to help those responsible for managing customer complaints, and to improve the overall quality of the provided customer service. The use of the system was illustrated by two cases: tracking and photographing bus stops, and

tracking and photographing recycling areas. Both cases utilized cameras installed in vehicles and location data. Furthermore, this paper discussed the issues faced in the design and implementation of the use cases. Based on a brief discussion with the companies, the initial reaction towards the prototype applications was positive, and the system was seen as an improvement over the previously utilized manual data collection. Still, to fully assess how the system contributed on the improvement of the customer complaint validation process, a more in-depth study would be required.

In any case, the use cases show that mobile platforms can work as a quick starting point for rapid prototyping – documentation and examples are easily found and the devices contain a number of built-in sensors. The disadvantage of mobile platforms is the lack of options for remote management, and single-board computers (e.g., Raspberry Pi) could provide a better platform if remote access is required. Unfortunately, it can be more challenging to find applicable examples and documentation when compared to commonly used mobile platforms. Additionally, both cases highlighted the importance of environmental factors – such as the availability of electricity, telecommunications, and installation of the prototype – even in cases when the primary goal of prototyping is in software testing or running short trials. The importance is seen especially when the testing is done in a real environment and should not disrupt the daily operation of the participating companies.

REFERENCES

Ian F. Akyildiz, Su Wy, Yogesh Sankarasubramaniam, and Erdal Cayirci. 2002. Wireless sensor networks: a survey. Computer Networks 38, 4 (Mar 2002), 393–422.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. Computer Networks 54, 15 (Oct 2010), 2787–2805.

Foreca. 2018. Havaintohistoria. Website. Retrieved May 14, 2018 from https://www.foreca.fi/Finland/Pori/havaintohistoria

Jere Grönman, Petri Rantanen, Mika Saari, Pekka Sillberg, and Juha Vihervaara. 2018. Low-cost ultrasound measurement system for accurate detection of container utilization rate. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE.

Ahmad Iftikhar, Petri Rantanen, Pekka Sillberg, Jorma Laaksonen, Shuhua Liu, Thomas Forss, Aqdas Malik, Marko Nieminen, Rakshith Shetty, Satoru Ishikawa, Jarno Kallio, Jukka P. Saarinen, Moncef Gabbouj, and Jari Soini. 2018. VisualLabel Integrated Multimedia Content Management and Access Framework. Information Modelling and Knowledge Bases XXIX (2018), 321 – 342.

Hannu Jaakkola, Jaak Henno, Tatjana Welzer Družovec, Bernhard Thalheim, and Jukka Mäkelä. 2016. Why information systems modelling is difficult. CEUR Workshop Proceedings 1677 (2016), 29–39.

R.C. Jisha, Aiswarya Jyothindranath, and L Sajitha Kumary. 2017. IoT based school bus tracking and arrival time prediction. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 509–514.

SeokJu Lee, Girma Tewolde, and Jaerock Kwon. 2014. Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application. In 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE, 353–358.

Mika Saari, Ahmad Muzaffar bin Baharudin, and Sami Hyrynsalmi. 2017. Survey of prototyping solutions utilizing Raspberry Pi. In 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 991–994.

Mika Saari, Ahmad Muzaffar bin Baharudin, Pekka Sillberg, Petri Rantanen, and Jari Soini. 2016. Embedded Linux controlled sensor network. In 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 1185–1189.

Mika Saari, Pekka Sillberg, Petri Rantanen, Jari Soini, and Haruka Fukai. 2015. Data collector service - practical approach with embedded Linux. In 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, 25-29 May 2015, Opatija, Croatia (International convention on information and communication technology, electronics and microelectronics). IEEE, 1037–1041.

PUBLICATION V

Reducing Energy Consumption with IoT Prototyping

Saari, M., Sillberg, P., Grönman, J., Kuusisto, M., Rantanen, P., Jaakkola, H. and Henno, J.

Acta Polytechnica Hungarica 16.9, SI (2019), 73–91
DOI: 10.12700/APH.16.9.2019.9.5

Reducing Energy Consumption with IoT Prototyping

Mika Saari*, Pekka Sillberg*, Jere Grönman*, Markku Kuusisto*, Petri Rantanen*, Hannu Jaakkola*, Jaak Henno**

*Tampere University, Faculty of Information Technology and Communication Sciences, Pohjoisranta 11A, 28101 Pori, Finland, mika.saari@tuni.fi, pekka.sillberg@tuni.fi, jere.gronman@tuni.fi, markku.kuusisto@tuni.fi, petri.rantanen@tuni.fi, hannu.jaakkola@tuni.fi

**Tallinn Technical University, School of Information technologies, Ehitajate tee 5, 19086 Tallinn, Estonia, jaak.henno@ttu.ee

Abstract: Nowadays, energy consumption and especially energy saving, are topics of great importance. Recent news regarding global warming has increased the need to save energy. In Finland, one of the major sources of energy consumption is housing. Furthermore, the heating of residential buildings accounts for up to 68% of housing energy consumption. Therefore, it is not surprising that apartment energy consumption and ways to save energy in housing are a popular research topic in Finland. In this paper, two different research areas are introduced: First, a literature survey is presented on the research subjects of energy saving in the area of real estate and housing. The goal is to gain overall knowledge of the current state of energy saving research. The overall conclusion is that knowledge of energy consumption improves efforts toward energy saving. Second, rapid prototyping with off-the-shelf devices and open source software are described. These devices are cheap to install, and a wide range of sensors are available. Consequently, it is important to deal with these topics together. The former studies provide knowledge about the usage of open hardware, open software, and open architectures with the development of prototype systems for gathering data. The literature survey gives us new information on the specialties of energy consumption measuring, offering a new area for modeling and developing prototype systems. These experiences will be taken forward and utilized in energy saving and environmentally sustainable solutions, such as Green Computing.

Keywords: IoT; Prototyping; Energy saving

1 Introduction

In the modern world, energy saving has become an important issue, in almost every aspect of life. Global warming is forcing people to search for low-energy solutions. It is important to be aware of the living comfort when thinking about the

low-energy solutions. For example, most people want the living temperature to be comfortable - not too low and not too high. Furthermore, the awareness of one's energy consumption has been proven to reduce overall energy usage. Thus, in the context of this paper, the research problem can be formulated as:

How to reduce energy consumption by collecting and serving suitable data?

For this problem, we are looking for a solution for two questions.

1) How to categorize the energy consumption related studies?
2) How to utilize free and open solutions in the energy consumption context preserving adequate living conditions?

In our use cases, we are especially looking for solutions that utilize open-source components and open hardware, architectures and interface specifications. This study belongs to the Internet of Things (IoT) research area and to studies focusing on Wireless Sensor Networks (WSN). In addition, one of the focus areas of this paper is rapid prototyping in the IoT world by using off-the-shelf devices. An example of rapid prototyping method was described by [1] for the automotive industry.

This paper introduces the application architectures and system models for IoT prototyping. Furthermore, sensors and sensor networks that collect data into the cloud are discussed, and more specifically, wireless sensor network (WSN) systems that can be utilized in testing data collection in rapid prototyping are of interest. In our use cases, the prototypes are built using off-the-shelf devices and tools. Additionally, Green ICT (Information and Communication Technology) should be part of the developing process when either the goal is to save energy or make systems which help to save energy.

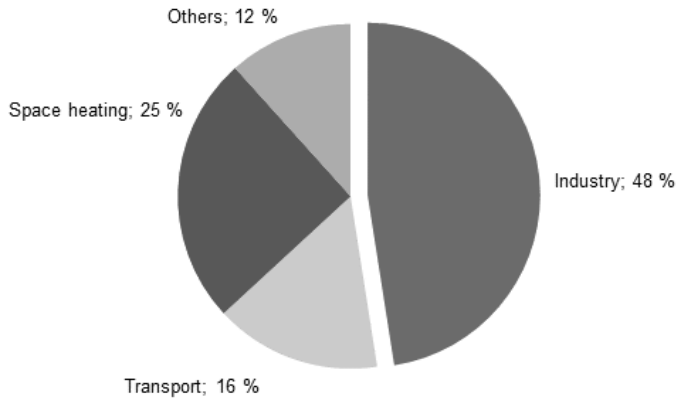

Figure 1
Finland's energy consumption by sectors in 2018 [2]

According to official statistics, collected and published by Statistics Finland, energy in Finland is produced mostly in three ways: wood, oil, and nuclear fission. These three sources combined add up to 66% of the energy produced in Finland. Various other sources of energy production include but are not limited to: coal, gas, water, peat, and wind. In their report, Statistics Finland [2] profile the Finnish energy consumption as shown in Fig. 1: Industry uses the most energy (48%) while heating comes in second place with 25% of energy consumed. Traffic is also a major consumer with 16% of the total energy used in Finland. Other sources then add up to the remaining 12%.

This research is focused on Finland (and further applicable in other northern countries), in which energy is often used for heating, instead of cooling (as is common in many other countries). The *"Cold weather raised energy consumption in housing in 2016"* report by Statistics Finland [3] shows that heating residential buildings consumed 46 TWh of energy in Finland during 2016. Furthermore, the heating of residential buildings was reported to account for up to 68% of the total energy consumption of housing with the second largest consumer of energy being heating water, accounting for 15%. Other notable energy consumers in Finnish households were electrical appliances, saunas, and lighting. The most common source of energy for heating was electricity, at 34%. The next most common source of energy was district heating (29%) and the third most common heating energy source was wood, at 22%, followed by heat pumps, at 9%. The usage of heat pumps in Finland has grown significantly since the start of the millennium because of their efficiency, saving energy and money compared to direct heating sources. All together, these four sources of energy made up about 95% of the energy used for heating in Finland. The remainder was mostly heating oil at approx. 5%, with other technologies accounting for less than 1%.

Our former research focus has been IoT and prototyping. This preliminary research will show how existing studies could be applied to a new research area. The structure of paper follows the research process: Section 2 includes a brief introduction to studies related in energy consumption. Section 3 continues with further analysis and categorization of energy consumption papers. Section 4 will present our studies and those findings, which could be combined with energy consumption monitoring. Further, the combined ideas of reducing energy consumption and prototype developing are introduced. Finally, Section 5 concludes the study.

2 Related Studies in Energy Consumption

This section deals with studies related to energy consumption. One important point of view is the awareness aspect of energy consumption. In [4], it was found that dormitory residents reduced electricity consumption when exposed to real-

time visual feedback and incentives. This study examined electricity and water usage. In the study, two dormitories were equipped with automated monitoring systems that provided high-resolution, real-time feedback. The study showed that the residents' awareness, knowledge, and behavior regarding energy saving improved after they were provided with relevant information and exposed to campaigns.

The study [5] examined the effects of energy saving, by analyzing the changes in the awareness and behavior of apartment residents after the promotion of energy-saving activities and their proper usage, and the provision of relevant information. In this study, the questionnaire included topics such as energy awareness and the knowledge and practice of energy conservation. In addition, this study performed an additional survey, which was conducted for women who were given energy-saving information and asked to participate in energy-saving activities after submitting the initial questionnaire. The results showed that energy-saving behavior improved after being provided with relevant information.

In the third study [6], the focus was on the meaning of comfort and comfort practices, barriers to and motivators for saving energy, and knowledge about the heating system. Data were collected from social housing tenants and university staff using surveys, interviews, and monthly energy meter readings. This study showed that warmth was mentioned most often as the meaning of comfort. In addition, comfort practices were to a large extent defined as temperature-related actions that were low in energy consumption. This study also found that willingness to change behavior was the greatest when the motivation was to save money.

The study [7] focused on energy-saving awareness, by using In-Home Display (IHD) devices. These devices provide real-time data about the use of electricity in specific appliances. Also, the costs of these devices were shown, and the users had the opportunity to reduce their electricity consumption. The result of this study was that the direct feedback provided by IHDs encouraged consumers to make more efficient use of energy. In addition, active IHD users were able to reduce their electricity consumption by about 7%, on average.

All these studies show that knowledge of energy consumption improves efforts toward energy saving.

3 Literature Survey

The introduction posed the research question: how to categorize the energy consumption related studies? To answer this research question, a literature review was performed, in order to map the existing knowledge in this domain.

3.1 Research Approach

The literature review used the Systematic Literature Review (SLR) method for collecting relevant primary studies and followed the guidelines given by Kitchenham and Charters [8]. For the SLR, an electronic literature search was executed. The databases used were IEEE Xplore Digital Library (IEEE) and Google Scholar. The survey was started by using the main search term: "Energy consumption". During the pilot study and related research [4-7], several other research terms arose such as "Temperature comfort", "Learning temperature comfort", "Apartment temperature comfort", "Smart home communication", "Real-time energy consumption monitoring," and "Energy apartment sensor". With a combination of these keywords, a good coverage of potential studies was obtained. The target amount of related studies was a total of fifty publications, as this amount would provide enough information for categorization and determination of research trends. Of these fifty publications a small number of papers were selected, which were considered to include the most relevant papers for the energy consumption or energy savings.

3.2 Categories for Existing Studies

To get an overview of the existing studies, the papers included in the study were analyzed for common topics. Most of the papers were relatively distinctive in terms of research objective, methodology, and application. Ultimately, based on the analysis of the research papers, we selected four categories taking into consideration the variations in research themes. The reason for choosing a relatively small amount of categories was to enable the examination of the details of research papers falling under the same category systematically. Selecting too many categories would have made it difficult to compare the trends or research methodologies. It is worth noting that some of the papers could be classified into more than one category. The research categories identified from the source material are:

- Comfort
- Retrofitting
- Network APIs
- IoT

The categories are listed according to the importance of the background research. The category ‘Comfort’ contains studies that discuss the basic elements for living comfort, which are often considered to be more important than energy saving. In general, comfort is an important aspect of energy saving. Too much saving means that the comfort of the living environment, such as thermal comfort and humidity, decreases. The most important factor is thermal comfort, which is taken into

account in several studies [9-14] in this category. Most of the research addressed previous studies, but [9] in particular reviewed thermal comfort research work and discussed the implications for the energy efficiency of buildings.

In our use cases, focus is on existing building stock and therefore the 'Retrofitting' category contains the research on applications or solutions installed in existing buildings. A different approach is used for monitoring energy consumption monitoring in new buildings and old buildings. In new buildings, monitoring applications and systems are included in the design phase of the building. For example, the heating system could be selected by weighing up the energy aspects. In old buildings, the main structure (e.g. the heating system) already exists, and the monitoring must fit this structure. This category consists of studies [15-20] where the presented application or solution was installed in existing buildings.

The study [15] focused on the problems of buying or renting a house. The potential purchaser or renter of the property does not know its living comfort factors such as temperature and lighting. This study introduced IoT sensors for the evaluation of the comfort levels of real estate properties. Another study [16] focused on studying and determining the cost-optimal renovation measures to decrease both the supplied and primary energy consumption of the building. This study encouraged apartment building owners to conduct thorough renovations toward nearly zero-energy apartment buildings.

The third category focuses on Application Programming Interfaces (APIs) and other methods that allow remote control or management of devices over networks. In addition, devices including a network API can provide (web) services usable by application developers or by client devices. A RESTful API is an architectural style for communications used in web service development, which was mentioned in [21] although the usage was not described in detail. The second study [22] present four RESTful services: one developed in Arduino and three mobile applications. A third study [23] integrated smart power outlets into the web and facilitated the development of extensions and novel features. They were implemented in a web user interface and a mobile phone interface for demonstration purposes. In addition, this was confirmed with a 12-month pilot deployment.

The study [24] described the construction of a smart outlet network as a system for automated energy-aware services utilizing humidity, temperature and light sensors, and motion sensor data. The sensors were installed on smart outlets and the appliances were under policy-based automatic control. This study also presented the deployed system in real-life environments.

The last category, 'IoT', includes the studies which do not fit in any of the other categories, but are nevertheless related to our focus area. This category is the widest and most of the papers could be included in it. Therefore, this research only introduces studies which: (*i*) collect the data in some way; (*ii*) save the data; and (*iii*) the saved data are then used or processed.

The survey [25] explored state-of-the-art control systems in buildings. The ref. [26] focused on intelligent control systems for energy and comfort management in smart energy buildings. The study [27] presented the wireless, smart comfort sensing system that they developed. This system consists of sensor nodes, which send data to a sink node that sends data to a PC. Another, lower-cost implementation was presented and discussed in [28], describing the hardware IoT infrastructure providing real-time monitoring in multiple school buildings. The sensor nodes and gateway node were based on Arduino boards or similar. A further study [29] also used low-cost devices in their HVAC and sensor system. IoT is also discussed in several studies [10], [13] and [21], which have been mentioned above.

Table 1
Breakdown of the papers reviewed

Category	Selected Studies	Author's Studies	Number of Studies
Comfort	[9-14]	[30]	7
Retrofitting	[15-20]		6
Network APIs	[21-24]	[30-36], [38]	12
IoT	[10], [13], [21], [25-29]	[30-31], [33], [37-40]	15

The results of the literature survey and the selected categories (Comfort, Retrofitting, Network APIs and IoT) can be seen in Table 1. The table also shows how the authors' own contribution related to the categories.

4 Prototype Systems and Models

This section gives a brief summary of our earlier studies related to rapid prototype development. The proof-of-concept demonstrations and prototype applications have been developed to illustrate how to utilize cost-effective, open, and modular solutions. The studies have been chosen based on their potential for including methods or technologies that could be transferred or exploited in the energy consumption monitoring or energy saving context.

4.1 Rapid Prototyping

In the context of rapid prototype development (and in the context of IoT devices in general), a working solution for gathering data needs:

- Hardware – a device or devices running the software
- Software to work with the data – collect, save, and transmit
- Technologies–choosing the right technologies for a use case makes things easier for both the developer and the user.

In our use cases, the prototype development has had more of software than hardware orientation. Data are gathered with embedded software, which controls the action of sensor devices. The data transfer to the cloud can be made in various ways and requires applicable software to control the sending and receiving of data. The WSN and sensor networks have several possible technologies for data transfer, for example: Ethernet [30], WiFi, ZigBEE [31] and LoRa [32]. In addition, power saving algorithms for WSN [33] and network topology related issues of Portable Fog Gateways [34] can be considered important topics.

The prototype systems gathered data which was saved to cloud-based services. In a basic example, the cloud service could be implemented with a Linux-based server and database [35], which has been modeled in [36].

Software development was carried out in several areas: data gathering software, data processing software, visualization of results, etc. The software development consisted of small-to-medium sized applications written in C/C++, Java, JavaScript or Python. The operating systems were generally chosen from the Open Source selection. For example, the Raspberry Pi is usually equipped with Linux–based operating systems (e.g. the Debian-based Raspbian). Also other software, such as databases, communication and web server software, was typically Open Source software.

Hardware development can be an integral part of prototype system development, but in our use cases the prototypes used off-the-shelf devices. In the past few years the price of microcontrollers, small computers and sensors has become much lower. At the same time, more and more features have been added to the off-the-shelf devices. These factors have made utilizing off-the-shelf devices both cheaper and easier, and it has also reduced the need to construct (or design) sensor or device packages from the ground-up using basic electronic components. Often used off-the-shelf devices include:

- Smartphones and tablets
- Single-board computers: Raspberry Pi, Beagle Bone, Intel Galileo, etc.
- Single-board microcontroller: Arduino Uno
- Sensors: Heat, humidity, pressure, movement, position, etc.

Using these off-the-shelf devices for the manufacturing and up scaling the number of prototype devices is more rapid than implementing a prototype based on printed circuit board design. In addition, the Raspberry Pi has been shown to be good choice for research projects and is a widely used device [37].

Furthermore, nowadays mobile phones have the ability to act as sensor devices. Even the basic Android smartphone has several of the following sensors: light, proximity, camera, microphone, touch, position (GPS, WiFi, Cellular), accelerometer, gyroscope, pressure, temperature, humidity. The data collection and processing can be handled in a smartphone. In addition, a basic smartphone

usually has more than adequate communication features: Bluetooth, WiFi, GSM, GPRS, 3G, 4G, etc. are often available.

4.2 Data Gathering with Sensor Network–Modeling, Piloting, and Testing

The sensor networks can be modeled as is illustrated in Fig. 2 [31]. The sensor nodes gather data and send it without processing to the master node. The master node may validate the received data, it may also process it, and send the data to the cloud. The data are usable from the cloud for various purposes.

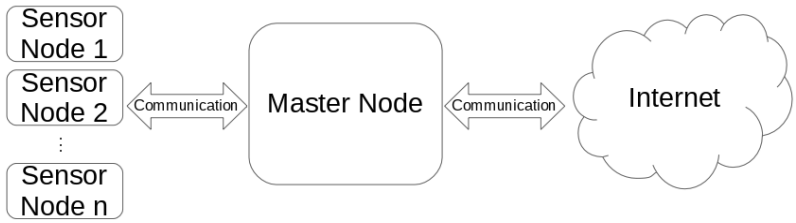

Figure 2
Basic model of sensor network [32]

This model was tested during the study [31], and a proof-of-concept solution was implemented and presented. Based on a survey of prototyping solutions that utilize Raspberry Pi the commonly used solutions were observed to adhere to this basic model even when no specific model was described in the studies [37]. However, the model shown in Fig. 2 has to be modified if smartphones are used as sensor nodes. Fig. 3 shows a combined presentation of the sensor node and master node model.

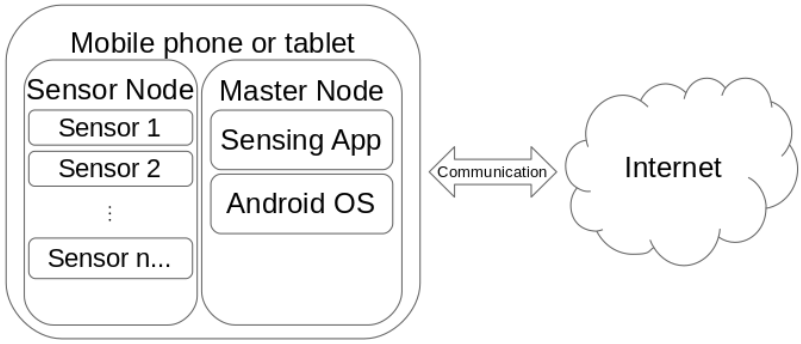

Figure 3
The combined sensor node—master node model for data gathering [36]

The model presented in Fig. 3 was developed especially for data collection with smartphones. The smartphone includes the necessary sensors, data storage, and communication channels for the data gathering prototype system. In addition, the Android operating system (OS) was used, which has enough capabilities to gather and store data. Also, the commonly used communication protocols are directly supported by the APIs provided by the OS. [36]

The studies [30-31] [36-37] show several important results:

- Study [30] introduced an example of how a cost-efficient single-board computer (SBC) can be used to gather sensory data, and how this data can be provided to the client over the public Internet. In addition, the use of standard protocols makes development easier, but not all development boards support all standards (in this case the I2C protocol).
- Study [31], mentioned that master nodes often have access to a constant power source, but one should carefully choose which components to use in remote sensors to minimize power consumption. In addition, most of the energy is consumed in the wireless transmission of data and consequently it is important to only send what is required (optimization of the nodes). The energy consumption issue was handled more specifically by [33].
- The survey about prototyping with Raspberry Pi was introduced in [37]. This paper shows that there is a lack of formalized approaches, methods, and tools in the research studies. Often only a single use case and a single system are described in the paper with a minimal use of testing practices and methods. The commonly used testing methods are software testing, software performance testing, and validation of data tests.

The conclusion from the results of the papers [30-31] [35-36] is that rapid prototyping with off-the-shelf devices is possible, but requires guidelines that include an architecture model of components—both software and hardware.

4.3 Prototype System: Road Condition Analysis and Visualization

Nowadays, almost everyone has a mobile phone and even the most basic smartphones often come embedded with a variety of sensors. In [38], smartphones were utilized to collect road condition data. The smartphone application developed during the research collects data from the phone's built-in sensors. The application can be installed in a common Android smartphone. This collected data could be further refined into more specific data, such as reports of bumps in the road, uneven road surfaces, roadworks, and so on. The data are sent to the cloud where they are processed. Fig. 4 shows the visualization of the captured data and the routes where the data were collected.

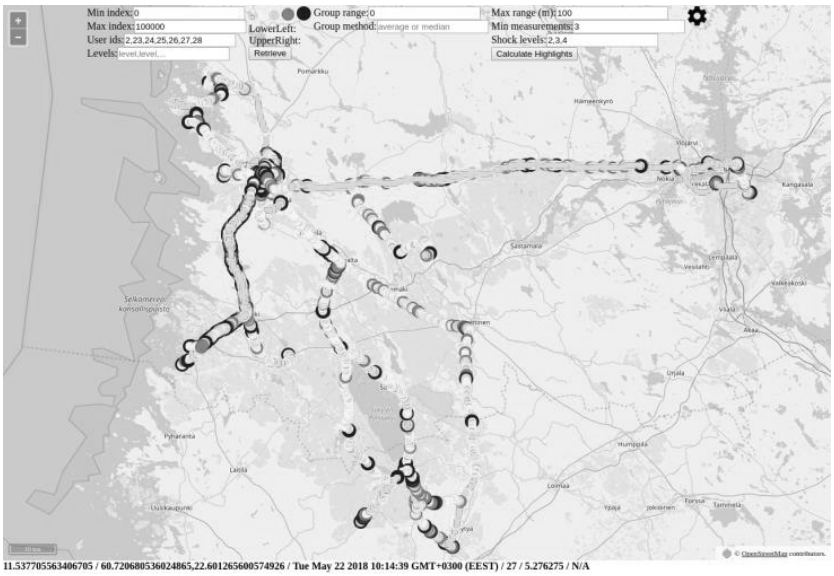

Figure 4
Visualization of the routes driven [38]

This research shows that it is possible to use a commonly used consumer product for data collecting. However, it turned out that, even though modern smartphones/devices are fairly similar by functionality, hardware differences can cause unexpected problems for implementation. Further, the embedded sensors are often not "calibrated" across devices and manufacturers. This can cause variances in the results and therefore comparison of data can be difficult if accuracy is of high concern. In addition, non-system-related effects and interference (environmental factors) may affect the final results e.g., when measuring shocks or vibrations different vehicles provide slightly different results. In addition, it is often necessary to perform pre-processing and filtering on-device, versus a fully service-implemented analysis.

A further result of this research is connected to the visualization of collected data. This is often no minor issue when measuring the quality of the user experience. Also, a fluent execution of visualization of a large dataset can be challenging, especially on a web browser.

4.4 Prototype: Approach to Image Data Collection

Customer complaints can be resolved by means of image and data collection. The research [39] introduces two prototypes installed in vehicles and a cloud service for autonomous collection of data. The first prototype—an Android application—

was implemented for a smartphone to take pictures of a bus as it approaches the bus stop. The second prototype was implemented for the Raspberry Pi single-board computer by using off-the-shelf devices such as a camera, GPS sensor, and 3G/4G wireless modem. The prototype was installed in a garbage truck to take pictures of recycling areas, as shown in Fig. 5.

Figure 5
Three pictures taken of a recycling area. Left: in daylight; center: at night; right: blocking obstacle [40]

The prototypes use a camera and GPS. The collected data—picture, location, time, etc.—were sent to the cloud server. The paper [39] discusses the differences and challenges faced in designing and implementing the two prototypes for different platforms.

The main conclusions were that mobile platforms (i.e., smartphones, tablets) can work as a quick starting point for rapid prototyping. These have embedded sensors, proper documentation, and the availability of examples, all of which support rapid prototyping. On the other hand, small computers like Raspberry Pi and microcontrollers offer a better option for use cases requiring remote management. Of course this has disadvantages, such as requiring more "hands-on" labor, and being more difficult to find examples or production quality code. In addition, both mobile platforms and small computers highlight the importance of environmental factors—such as the availability of electricity, telecommunications, and installation of the prototype [39].

4.5 Prototype: Counting Passengers from Image Data

The research [40] was the result of a real-life need for counting passengers. In the summer of 2018 a large public event was organized in the city of Pori, Finland. The event had free-to-ride buses and the organizer wished to collect statistics about the bus passengers: Where they got in and where they got out. The use case utilized cost-effective and off-the-shelf components such as the Raspberry Pi 3 computer, position sensors, and cameras. In this use case, the software used was Open Source Computer Vision Library version 3.

Figure 6
An example of the detection area of the bus, as seen by the device [40]

During the research, a prototype system was developed, consisting of hardware and software components. The prototype takes pictures, as shown in Fig. 6. The pictures are processed by the system, which was based on image analysis and shape detection. The data are processed in the Raspberry Pi and the results of the processed data are sent to the cloud server. [40]

4.6 Toward Reducing Energy Consumption with IoT Prototyping

An important part of achieving energy usage reductions is a reliable way of collecting data about current environmental conditions. The research presented in this section (Section 4) illustrated simple models that could be used when implementing a sensor network for collecting data. Furthermore, Section 4.2 illustrated certain pitfalls related to currently used approaches and highlighted the lack of existing model for rapid prototyping in the IoT domain. Sections 4.3 and 4.4 showed advantages of using smartphones as tools for data collection. Modern smartphones contain a huge variety of built-in sensors and the available devices range for low-cost affordable models to more expensive high-end devices. Today, almost everyone already has a smartphone, and thus, the cost of using smartphones for environmental monitoring can be negligible. Additionally, even the low-end devices are capable of running simple applications, that can be used to show statistics about current living conditions, and at least in theory, to provide the user with interfaces for controlling the environment. Unfortunately, there are challenges related to installing devices to real-life scenarios, such as, creating solid, durable packaging for the sensors and the availability of electricity and

telecommunications. Specifically when dealing with rapid prototype development and actual locations, there can be unexpected challenges, even when not considering the interoperability issues with existing structures and systems. More advanced scenarios can be realized with customizable devices. Section 4.5 described how Raspberry Pi could be used to monitor passenger ridership, an approach that could be easily expanded to energy consumption domain. Detecting whether rooms or buildings are occupied can have huge effect on the cooling and heating requirements. Furthermore, all of the presented prototypes use free and open software and low-cost modular components proofing that rapid prototyping with off-the-shelf devices is possible.

Conclusions

One of the initial research questions for this study was "How to categorize the energy consumption related studies?" Based on the literature review carried out, the existing studies can be roughly divided into four distinct categories: studies related to measuring and ensuring occupant comfort in buildings; research on how to extend existing systems with modern sensor and optimization solutions (retrofitting); studies on the usage and description of network-based APIs; and studies on IoT-based devices in general. All of these categories—comfort, retrofitting, network APIs, and IoT—include a wide array of existing research and provide numerous examples of applications and systems for monitoring and optimizing energy consumption. Several conclusions can be drawn from the results of the literature review, and from our previous experience in prototype development in the various research projects presented in this paper.

Our second research question was "How to utilize free and open solutions in the energy consumption context preserving adequate living conditions?" In the scope of this paper, the solutions for this question answered more on the basic technical problems. The paper gave insights on available software and hardware options, but the aspect of preserving living conditions was given less focus, and would require more extensive research.

In existing studies, IoT often consists more of "proof-of-concept" style research. The studies present a use case, various testing methods, and results, but often no formal model for testing or benchmarking is described. Without further studies it is difficult to say why there is an apparent lack of a standardized or *de facto* model for rapid IoT prototype development, but research on developing such a model or applying an existing model for the IoT context could be one potential direction for future studies.

Mobile devices (i.e., smartphones, tablets) can work as a good starting point for prototype development—they are ubiquitous, and they come embedded with various built-in sensors. Documentation and application examples are, in general, easy to find, and the utilization of mobile devices can be combined with off-the-shelf devices to create more complex systems. Off-the-shelf products—such as the Raspberry Pi single-board computer and wide multitude of available sensors—

have become much cheaper in recent years and offer adequate performance with a relatively good set of features and expansion capabilities. The market has also seen an increase in cheaper commercial sensor products targeted at consumers (end users). This price and market development has caused an increase in research utilizing cost-effective off-the-shelf devices as opposed to building and designing devices (e.g., sensor nodes) from the "ground up". Additionally, the increase in commercial products has enabled people with lesser technical knowledge to buy and set up sensor devices in their homes. Unfortunately, the interoperability of existing systems (air conditioning systems installed in older buildings, commercial products lacking proper interfaces or APIs, etc...) is often less than seamless and connecting the systems to available off-the-shelf devices can be challenging. With more barebone devices (Raspberry Pi, Arduino, etc.), packaging, designing a case, and installing the sensor node in a real-life environment or for outdoor use can pose further difficulties.

Finally, the paper attempted to answer the question: “How to reduce energy consumption by collecting and serving suitable data?” Based on the existing studies, the availability of energy consumption information can have a huge effect on people's habits, and properly presented usage statistics can lead to energy savings. In existing systems, the information is often limited to simple statistics (numerical details, graphs). Unfortunately, meaningful visualization can be challenging: How to select what is "meaningful"? How detailed should the statistics be? And how should the information be presented? In some cases, the user cannot affect the energy consumption and occupant comfort as desired. The user may not have access to the building's air conditioning or the building may not have devices capable of altering the indoor air quality (i.e., CO_2 levels, humidity, temperature, etc.)—should these statistics still be shown to the user? Furthermore, a building seldom has only a single occupant, and taking the possibly conflicting preferences of the users fully into account may in practice even be impossible. One potential research topic could be how to tackle the aforementioned issues, perhaps by utilizing A.I. or modern smart devices.

Acknowledgements

This work was supported by the European Regional Development Fund. These results were used when planning the ongoing “KIEMI” research project by Tampere University.

References

[1] Dubar, I. G., Bogdan, R., & Popa, M. (2017) External rapid prototyping validation system for the automotive development cycle. Acta Polytechnica Hungarica, 14(6), 41-57, https://doi.org/10.12700/APH.14.6.2017.6.3

[2] Official Statistics of Finland (OSF): Energy supply and consumption [e-publication] ISSN 1799-7976. 4th quarter 2018, Helsinki: Statistics Finland [referred: 24.6.2019] Access method: http://www.stat.fi/til/ehk/2018/04/ehk_2018_04_2019-03-28_tie_001_en.html

[3] Official Statistics of Finland (OSF): Energy consumption in households [e-publication] ISSN 2323-329X. 2016. Helsinki: Statistics Finland [referred: 24.6.2019] Access method: http://www.stat.fi/til/asen/2016/asen_2016_2017-11-17_tie_001_en.html

[4] J. E. Petersen, V. Shunturov, K. Janda, G. Platt, and K. Weinberger, "Dormitory residents reduce electricity consumption when exposed to real-time visual feedback and incentives," International Journal of Sustainability in Higher Education, Vol. 8, No. 1, pp. 16-33, 2007

[5] N. N. Kang, S. H. Cho, and J. T. Kim, "The energy-saving effects of apartment residents' awareness and behavior," Energy and Buildings, Vol. 46, pp. 112-122, 2012

[6] G. M. Huebner, J. Cooper, and K. Jones, "Domestic energy consumption - What role do comfort, habit, and knowledge about the heating system play?" Energy and Buildings, Vol. 66, pp. 626-636, 2013

[7] A. Faruqui, S. Sergici, and A. Sharif, "The impact of informational feedback on energy consumption - A survey of the experimental evidence," Energy, Vol. 35, No. 4, pp. 1598-1608, 2010

[8] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3," EBSE Tech. Rep. EBSE-2007-01, 2007

[9] L. Yang, H. Yan, and J. C. Lam, "Thermal comfort and building energy consumption implications – A review," Applied Energy, Vol. 115, pp. 164-173, 2014

[10] D. Marković, D. Vujicic, Z. Jovanovic, U. Pesovic, S. Randik, and D. Jagodic, "Concept of IoT system for monitoring conditions of thermal comfort," in International Scientific Conference "UNITECH 2016," 2016, November

[11] M. Taleghani, M. Tenpierik, S. Kurvers, and A. van den Dobbelsteen, "A review into thermal comfort in buildings," Renew. Sustain. Energy Rev., Vol. 26, pp. 201-215, Oct. 2013

[12] F. Salamone, L. Belussi, C. Curro, L. Danza, M. Ghellere, G. Guazzi, B. Lenzi, V. Megale, and I. Meroni, "Integrated Method for Personal Thermal Comfort Assessment and Optimization through Users," Sensors, 2018

[13] L. Ciabattoni, F. Ferracuti, G. Ippoliti, S. Longhi, and G. Turri, "IoT based indoor personal comfort levels monitoring," in 2016 IEEE International Conference on Consumer Electronics (ICCE), 2016, December 2015, pp. 125-126

[14] A. Ghahramani, F. Jazizadeh, and B. Becerik-Gerber, "A knowledge based approach for selecting energy-aware and comfort-driven HVAC

temperature set points," Energy and Buildings, Vol. 85, pp. 536-548, Dec. 2014

[15] Y. Obuchi, T. Yamasaki, K. Aizawa, S. Toriumi, and M. Hayashi, "Measurement and evaluation of comfort levels of apartments using IoT sensors," IEEE International Conference on Consumer Electronics (ICCE), 2018, pp. 1-6

[16] T. Niemelä, R. Kosonen, and J. Jokisalo, "Cost-effectiveness of energy performance renovation measures in Finnish brick apartment buildings," Energy Build., Vol. 137, pp. 60-75, Feb. 2017

[17] B. E. Medina and L. T. Manera, "Retrofit of air conditioning systems through a Wireless Sensor and Actuator Network: An IoT-based application for smart buildings," IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), 2017, pp. 49-53

[18] S. R. West, J. K. Ward, and J. Wall, "Trial results from a model predictive control and optimisation system for commercial building HVAC," Energy and Buildings, Vol. 72, pp. 271-279, Apr. 2014

[19] S. Gupta, M. S. Reynolds, and S. N. Patel, "ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home," in UbiComp, 2010

[20] C. A. Björkskog, G. Jacucci, L. Gamberini, T. Nieminen, T. Mikkola, C. Torstensson, and M. Bertoncini, "EnergyLife: Pervasive Energy Awareness for Households," in Proceedings of the 12th ACM International Conference Adjunct Papers on Ubiquitous Computing - Adjunct, 2010, pp. 361-362

[21] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake, "Leveraging smart meter data to recognize home appliances," IEEE International Conference on Pervasive Computing and Communications, 2012, pp. 190-197

[22] L. Özgür, V. K. Akram, M. Challenger, and O. Dağdeviren, "An IoT based smart thermostat," 5th International Conference on Electrical and Electronic Engineering (ICEEE), 2018, pp. 252-256

[23] M. Weiss and D. Guinard, "Increasing Energy Awareness Through Web-enabled Power Outlets," in Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia, 2010, p. 20:1-20:10

[24] N. Morimoto, Y. Fujita, M. Yoshida, H. Yoshimizu, M. Takiyamada, T. Akehi, and M. Tanaka, "Smart Outlet Network for Energy-Aware Services Utilizing Various Sensor Information," 27th International Conference on Advanced Information Networking and Applications Workshops, 2013, pp. 1630-1635

[25] B. L. Risteska Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," J. Clean. Prod., Vol. 140, pp. 1454-1464, Jan. 2017

[26] P. H. Shaikh, N. B. M. Nor, P. Nallagownden, I. Elamvazuthi, and T. Ibrahim, "A review on optimized control systems for building energy and comfort management of smart sustainable buildings," Renewable and Sustainable Energy Reviews, Vol. 34, pp. 409-429, Jun. 2014

[27] A. Kumar and G. P. Hancke, "An Energy-Efficient Smart Comfort Sensing System Based on the IEEE 1451 Standard for Green Buildings," IEEE Sensors Journal, Vol. 14, No. 12, pp. 4245-4252, Dec. 2014

[28] L. Pocero, D. Amaxilatis, G. Mylonas, and I. Chatzigiannakis, "Open source IoT meter devices for smart and energy-efficient school buildings," HardwareX, Vol. 1, pp. 54-67, Apr. 2017

[29] S. Godo, J. Haase, and H. Nishi, "Air conditioning control using selfpowered sensor considering comfort level and occupant location," in IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society, 2015, pp. 002497-002502

[30] M. Saari, P. Sillberg, P. Rantanen, J. Soini, and H. Fukai, "Data collector service - practical approach with embedded linux," 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, pp. 1037-1041, IEEE

[31] M. Saari, A. M. Baharudin, P. Sillberg, P. Rantanen, and J. Soini, "Embedded Linux controlled sensor network," 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016, pp. 1185-1189, IEEE

[32] M. Saari, A. M. bin Baharudin, P. Sillberg, S. Hyrynsalmi, and W. Yan, "LoRa — A survey of recent research trends," 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, May, pp. 0872-0877, IEEE

[33] A. M. bin Baharudin, M. Saari, P. Sillberg, P. Rantanen, J. Soini, and T. Kuroda, "Low-energy algorithm for self-controlled Wireless Sensor Nodes," International Conference on Wireless Networks and Mobile Communications (WINCOM), 2016, pp. 42-46

[34] A. M. bin Baharudin, M. Saari, P. Sillberg, P. Rantanen, J. Soini, J. Jaakkola, W. Yan, "Portable Fog Gateways for Resilient Sensors Data Aggregation in Internet-less Environment," Eng. J., Vol. 22, No. 3, pp. 221-232, Jun. 2018

[35] I. Ahmad, P. Rantanen, P. Sillberg, J. Laaksonen, S. Liu, "VisualLabel: An Integrated Multimedia Content Management and Access Framework," in Proceedings of the 27th International Conference on Information Modelling and Knowledge Bases, EJC 2017, 2017, pp. 332-353

[36] P. Sillberg, M. Saari, J. Grönman, P. Rantanen and M. Kuusisto, "Interpretation, Modeling and Visualization of Crowdsourced Road Condition Data", IS-TRIA2019, Submitted

[37] M. Saari, A. M. bin Baharudin, and S. Hyrynsalmi, "Survey of prototyping solutions utilizing Raspberry Pi," 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017, pp. 991-994, IEEE

[38] P. Sillberg, J. Gronman, P. Rantanen, M. Saari, and M. Kuusisto, "Challenges in the Interpretation of Crowdsourced Road Condition Data," International Conference on Intelligent Systems (IS), 2018, pp. 215-221

[39] J. Grönman, P. Rantanen, M. Saari, P. Sillberg, and H. Jaakkola, "Lessons Learned from Developing Prototypes for Customer Complaint Validation," in CEUR Workshop Proceedings, 2018

[40] J. Grönman, P. Sillberg, P. Rantanen, and M. Saari, "People Counting in a Public Event—Use Case: Free-to-Ride Bus," 42th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019, IEEE

PUBLICATION VI

Framework and Development Process for IoT Data Gathering

Saari, M., Rantanen, P., Hyrynsalmi, S. and Hästbacka, D.

Advances in Intelligent Systems Research and Innovation. Ed. by Sgurev, V., Jotsov, V. and Kacprzyk, J. 2022, 41–60. (Extension of peer reviewed conference publication Saari, Rantanen and Hyrynsalmi, 2020)
DOI: 10.1007/978-3-030-78124-8_3

Framework and Development Process for IoT Data Gathering

Mika Saari, Petri Rantanen, Sami Hyrynsalmi and David Hästbacka

Abstract The Internet of Things (IoT) is a growing area in everyday life. New applications under the umbrella term IoT are being developed continually. This development has raised the need for framework definitions for different purposes. This research introduces a special software/hardware framework for data gathering systems to be used in IoT related systems. The purpose of the research is to show the usability of a certain software/hardware combination in prototype development. The software/hardware framework has been developed during several research projects by following the same prototype development process. This is proposed as a descriptive model for the prototyping process. The main contribution of this research is the framework itself. The framework consists of a model of the system with selected components. The placement of the sensor network is also presented. The purpose of the framework is to guide and assist the construction of data gathering prototypes. Furthermore, the advantages of the framework are to support re-usability, portability, and interchangeability. This research introduces the framework, its main components, and their interconnections. In addition, the prototype development process used is presented.

Mika Saari
Tampere University, Computing Sciences, Pori, Finland e-mail: mika.saari@tuni.fi

Petri Rantanen
Tampere University, Computing Sciences, Pori, Finland e-mail: petri.rantanen@tuni.fi

Sami Hyrynsalmi
LUT University, Department of Software Engineering, Lahti, Finland e-mail: sami.hyrynsalmi@lut.fi

David Hästbacka
Tampere University, Computing Sciences, Hervanta Campus, Finland e-mail: david.hastbacka@tuni.fi

1 Introduction

The Internet of Things (IoT) is a growing area in everyday life. New applications under the umbrella term IoT are being developed continually. The IoT paradigm is the integration of several technologies and communications solutions [1]. This development has raised the need for framework definitions for different purposes. For example, the draft of the IEEE standard [2] defines an architectural framework for the Internet of Things (IoT).

This article introduces a special software/hardware (SW/HW) framework for data gathering systems to be used in IoT related systems. The research question can be stated as follows: How to generalize the prototyping of IoT data gathering in a framework of required software and hardware components?

The research question was formulated during previous data gathering prototype system development projects. The main purpose of these prototypes is to gather data, for example environmental data such as temperature, humidity, or carbon dioxide levels. The aim was to focus on the reproducibility of components within the development process. This study presents guidelines for selecting the required software and hardware components. The purpose of the SW/HW framework is to guide and assist when constructing data gathering prototypes. Furthermore, the advantages of the framework are that it supports re-usability, portability, and interchangeability.

This study is part of the research related to the Internet of Things (IoT) carried out by the Software Engineering and Intelligent Systems (SEIntS) group at Tampere University, Pori. The SW/HW framework has been developed during several research projects. These projects have contained multiple iteration rounds. Many of these rounds have produced a research article, whose main target was to describe the working prototype. The first prototype system was introduced by Saari et al. [3] in 2015. That research introduced the initial idea of a framework and a working implementation from it. Research on reducing energy consumption [4] presented the advantages of rapid prototyping with off-the-shelf devices and open source software.

The main idea of prototype development has been to start with off-the-shelf devices and open source software. These key software and hardware components are then modified in the desired direction and usually a working prototype system is produced.

The main result from our research is the framework itself. This has the ability to act as a guiding principle when developing new prototypes for gathering data. This framework also aims to represent the development of software and hardware usage in data gathering systems; in particular the evolution in the usage of both software and hardware is considered in different parts of the system. The framework could be used as a model when planning new data gathering prototypes for sensor networks.

The second finding made during the research is a descriptive model for the prototyping process (DMPP). This is a model of prototype development practices that have been applied in several research projects between university and enterprises (mostly small and medium-sized enterprises (SMEs)) in Finland. The purpose of the model is to introduce how academic research can conduct prototype development with regional enterprises.

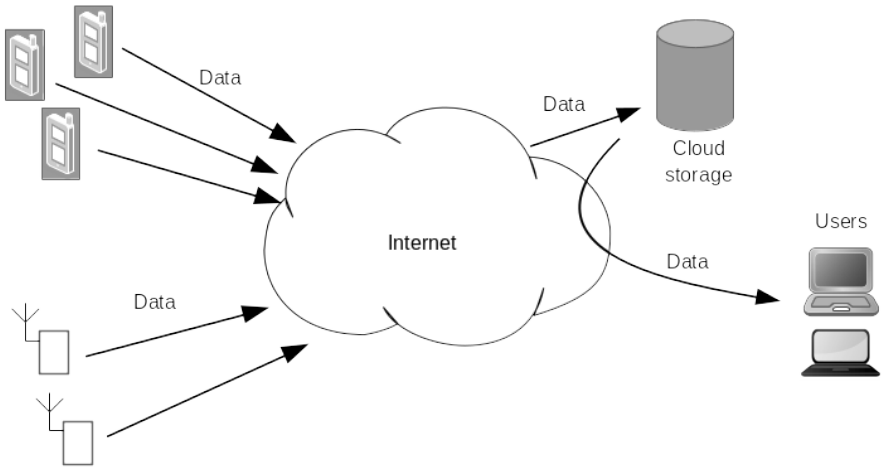

Fig. 1 The overall architecture of a data gathering IoT prototype system.

The structure of this paper is as follows: In Section II, we review the related research about prototyping, the development process, and related frameworks. In Section III, we introduce an implementation of university-enterprise collaboration in prototype development described by means of process modeling notation. Section IV introduces the SW/HW framework for IoT data gathering. Section V continues by describing the validation and testing of the framework. Section VI includes a discussion and suggestions for future research on the topic and finally, Section VI summarizes the study.

2 Related research

The importance of prototyping embedded SW/HW systems was introduced by [5]. The reason for this was because system differences had increased, and the product relied mainly on variations in software and system features. In addition, involving users in the specification process is important because more and more customers expect solutions and services tailored exactly to their particular needs. In a more recent study, [6] developed a working prototype using off-the-shelf components. This also showed that a lengthy product development life cycle is not required when using a rapid prototyping process.

Rapid prototyping could be presented as a circle (Fig. 2). Rapid prototyping includes three stages: making a prototype, reviewing the result, and refining and iterating [7]. We used the idea of rapid prototyping in our projects. The working prototype solution in the context of IoT requirements is as follows: hardware to run the software; software to collect, save, and transmit the data; the right technologies for the use cases to make things easier for both developer and user. [4]

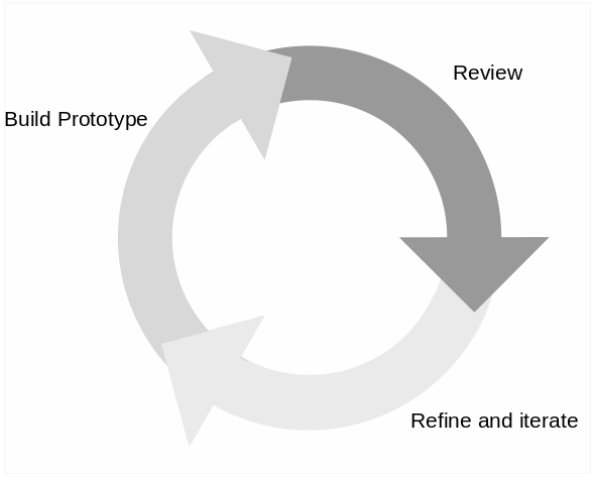

Fig. 2 The circle of the rapid prototyping process. Adapted from [7]

2.1 Development process of the prototypes

The IoT prototyping process can be viewed from two perspectives: a software development process and an embedded hardware development process. Furthermore, the authors have researched the IoT data gathering prototype development process by collecting data from several prototyping processes.

A development process from the area of software development is suitable for this study. The developed prototypes and the presented SW/HW framework include a lot of software development. The process model could be descriptive or prescriptive. A prescriptive model tells how the process should be performed, whereas a descriptive model tells how a process is performed in a particular environment. The third option, a proscriptive model, also describes the activities that could be done [8]. The descriptive process model (DPM) [9] [10] introduces an eight step approach for producing a process model. These steps are divided into two phases: the set-up phase and the execution phase. The eight step approach was used when the descriptive model for the prototyping process (DMPP) was developed. The DMPP is presented in Section 3.

The book "Introduction to Embedded Systems - A Cyber-Physical Systems Approach" by Lee and Sashia is based on the idea that designing and implementing an embedded system consists of three major parts of the process: modeling, design, and analysis [11]. The modeling phase specifies what a system does by defining the system model and the set of requirements. The artifacts, such as a combination of software and hardware components, are produced in the design phase. An artifact is a working system and it describes how the system works. In the analysis phase, information on the system is obtained to specify why the system works as it works. [11]

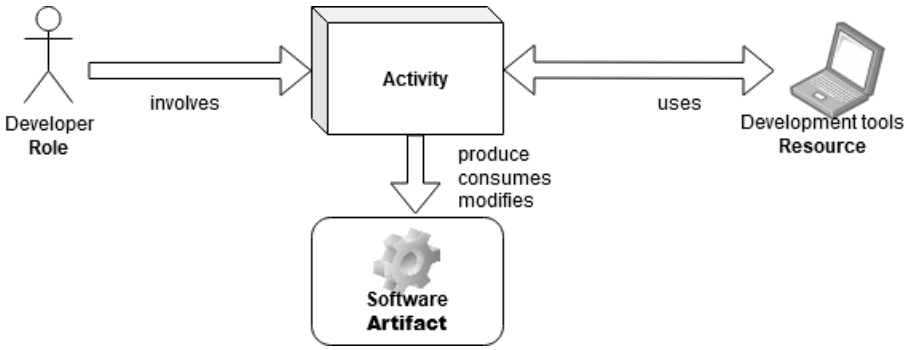

Fig. 3 Example of basic concepts related to the prototype development process. Adapted from [18]

2.2 The framework for prototyping

The SW/HW framework idea is not a new issue in the research field. For example, earlier studies [12], [13], [14] have addressed the framework subject from a real-time system perspective. In these studies, the design was at micro-controller level, whereas our prototypes use off-the shelf single-board micro-controllers. For example, Srivastava and Brodersen handled board level module generation, system software generation, and hardware-software integration in a unified framework [14]. Their study mentions a rapid prototyping method, but it was not explained further.

IoT architecture consists of several components, which can be divided into layers as follows: sensing layer, networking layer, service layer, and interface layer [15], [16]. The SW/HW framework is focused on the sensing and service layers. The networking layer exists but is not the focus of our research. The interface layer is described to the user in the SW/HW framework but is excluded from the study.

A wireless sensor network (WSN) can be used in various application areas. A WSN includes sensor nodes, which consist of sensing, data processing, and communicating components. A sensor network is composed of a large number of sensor nodes, which send data to the data storage. Since sensor nodes have data processing ability, the uploaded data can be either raw or pre-processed [17].

3 Descriptive model for the prototyping process (DMPP)

We introduced our descriptive software process model for IoT prototyping in [18]. The purpose of this section is to present how the selected process model has supported the development of the framework.

The DMPP [18] could be extended to include hardware, because the model itself does not limit the type of artifact. Therefore, when the process is mentioned in this study, it generally means every kind of artifact, i.e., software or hardware, made

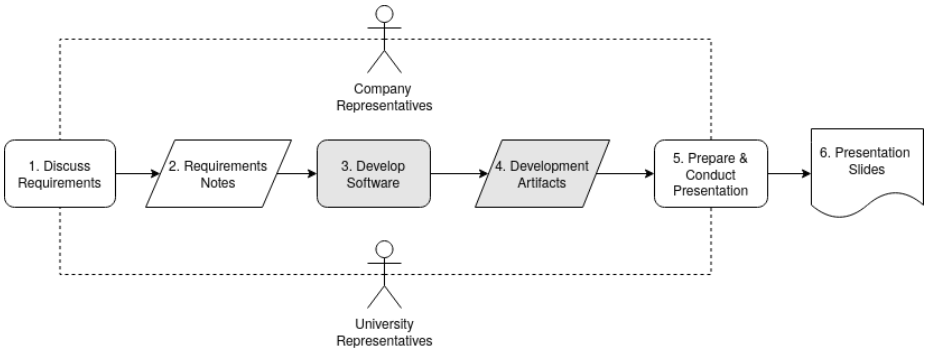

Fig. 4 Process model for prototype development. Adapted from [18].

in the prototype development process. The focus is on modeling the prototyping process in a research context, but its use in general is not restricted.

The DMPP was developed using the aforementioned descriptive process model (DPM) approach [10]. The basic concepts related to processes are role, activity, resource, and artifact. The example is illustrated by the developer (role) involved in software developing (activity) using a programming tool (resource). The activity produces some software (artifact) used in the prototype system. The process data for the model is collected through interviews with the developers involved in the four different prototype development processes. These four prototype development projects and their outcomes are reported in several studies [19], [20], [21], [22]. Common to all of the studies are that they present developed IoT prototype systems that gather data. Although the software and hardware components in the prototypes vary, overall they can be used to model the prototype development process.

Fig. 4 presents the developed DMPP [18]. The model includes six steps. These steps support or use the SW/HW framework in the following ways:

1. The first step starts from the requirements definition, a collaborative discussion between the developers and the client. The client defines what kind of data would be useful. The developer group starts to define the hardware and overall architecture of system and how the data will be collected by the software. The selected hardware mostly determines the software environment and tools used.
2. The outcome of the discussion is the first artifacts: for example, the prototype system requirements within the discussion notes. The developer group constructs the first architecture model of the component interconnections.
3. The third step is the software/hardware prototype development made by the research group including the project manager and software/hardware developers. The clients' representatives are involved in the development process in the role of instructor. In this step the SW/HW framework is used as the guideline for selecting the components for the developed prototype.
4. The fourth step introduces the working prototype artifact. It contains the developed software and hardware components. Also, the interconnections of the components are tested. The testing process overall is usually only the functional testing of the

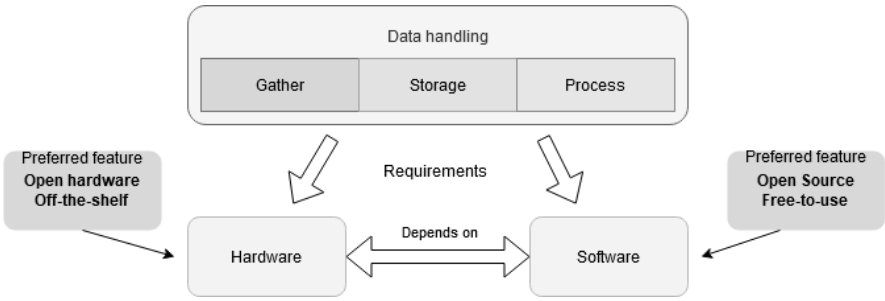

Fig. 5 Relationships of the SW/HW framework.

prototype system. Additionally, the gathered data are inspected and if possible, compared to the expected results.

5. The fifth step includes preparing the outcome of the development process. The SW/HW framework can be complemented if necessary.
6. The sixth step is to publish the results, for example, the prototype system, the collected data, and the analysis of the project.

The process model in Fig. 4 is a simplified presentation of the prototype development process and therefore it does not mention common procedures such as iterations, testing, and customer testing.

Iterations are an efficient way to test and develop an idea. The first working prototype is made as simple as possible with basic components. For example, the hardware could be chosen at first only for testing the idea. If the idea works, the hardware is changed for more suitable hardware in the next iteration round. A good example of this is the application where we tested the use case [23]: Is it possible to take a photo in selected GPS coordinates automatically and send this photo to the cloud storage? This idea was tested with the Android application in a smartphone and the idea was found to be a workable solution. The smartphone had some limitations with the automation: The developed photographing application had to be started, it was not possible to be aware of the application crashing, and the possibility of remote control was not easily implemented. The second iteration round to solve the same use case was carried out with the following hardware: Raspberry Pi, camera, battery, GPS sensor, WiFi, and 4G modem. This time the Raspberry Pi OS made it possible to implement the automated operations and remote control with Linux tools.

4 SW/HW Framework for IoT data gathering

In this section we introduce the SW/HW framework for an IoT data gathering system. The framework consists of several hardware and software components. The purpose and advantage of the framework is to support re-usability, portability, and

interchangeability. Another purpose of the framework is to guide and assist the construction of data gathering prototypes.

The definition of the SW/HW framework has been made in several academic research projects where the focus has been the collection of data using self-made prototype systems. Fig. 5 illustrates the framework and its relationships. The previously presented model in Section 3 has connections to the framework and its development. DMPP step 1 gives guidelines for data handling. The client tells the developers what kind of data are useful. In step 3 the developers make the decisions on what kinds of software and hardware components are needed to fulfill the clients' data gathering expectations.

Furthermore, Figure 5 clarifies the interconnections of the framework components. These interconnections guide the selection of software and hardware components when constructing a prototype. The framework relies on off-the-shelf devices, because this speeds up development by minimizing hardware design and implementation [24]. The information that is to be collected determines the collection and structure used by the software and hardware components. Optional features are also used when selecting the components. In hardware, open hardware and off-the-shelf devices are preferable, because they are reasonably priced, quickly available, and have community support. The software components should have similar features: open source and community support. Please note that these criteria are not preferred when implementing the final application for production use.

The first question for the constructor of an IoT data gathering system when implementing the prototype is what information or data to collect. The answer to the question should be clear and it should also be the motivator of constructing the system. The next three questions are presented after the decision to collect data:

- How to gather data?
- How to store data?
- How to process data?

For an end user or client the availability of the data is important. Therefore the preferable place to store and process the data is in some cloud storage or server. Cloud storage could be some old Linux server or fully optimized commercial cloud computing service. However, before the data are in cloud storage, they have to be collected and stored temporarily in a sensor device or gateway device. Furthermore, the first data processing and storing should be managed by the sensor device or gateway device. For example, if the network has a communication problem, there is the possibility of losing data unless the data are forwarded rather than being stored temporarily. The SW/HW framework focuses on data storage and processing in the sensor or gateway layers.

Fig. 6 shows the overall layer architecture of the IoT data gathering prototype system in a sensor network. The presented SW/HW framework is located in the lower part, containing the sensors, gateways, and preferable means of communication to operate with cloud storage. A few explanations of the Fig. are given below:

- The user and client layer utilizes the collected data.

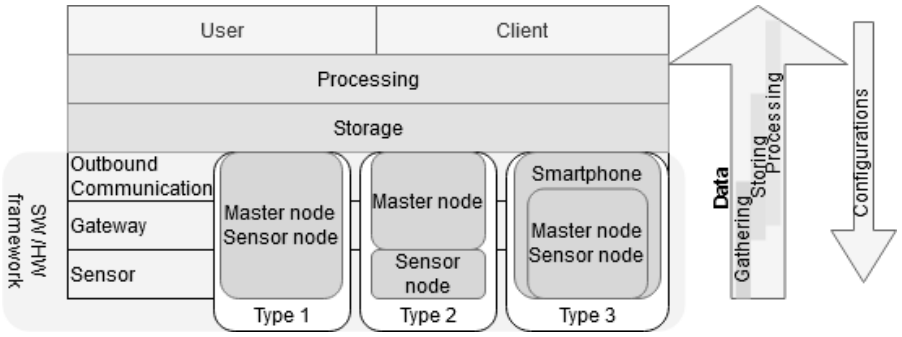

Fig. 6 Layer diagram of IoT data gathering and how the SW/HW framework is placed in it.

- A processing layer is needed in most use cases. The data are processed in the way the clients require. The user can use the raw, unprocessed data.
- The storage layer collects and saves the data. The purpose of this level is to ensure data retention.
- The outbound communication layer belongs to the SW/HW framework. Its purpose is to offer a suitable data transfer method.
- Data gathering has three different hardware constructions: Type 1 - master node-sensor node combination; Type 2 - several sensor nodes collect the data in one master node; Type 3 - Fully operational sensor device - a smartphone collects the data.
- The data flow from the sensors to the client or user. The data could be temporarily stored on the gateway level. The data processing could be also done on then gateway level if it is necessary and possible.
- Configurations and monitoring are enabled to ensure faultless operation and for testing purposes.

The division of the master node - sensor node into three different types of construction supports the versatility of the SW/HW framework. The main idea is that these three types make it possible to collect a wide range of data.

4.1 Hardware of the SW/HW Framework

The data gathering hardware can be divided on a higher level into three type of constructions, as can be seen in Fig. 6. The framework uses off-the-shelf hardware and devices. This limitation accelerates prototype development as at least partially tested devices can be used. The hardware can be categorized in two parts:

- Sensor node - Sensor hardware consisting of a combination of sensors and control device.
- Master node - data gathering and storing device that has the capability of collecting, storing, and processing data.

Table 1 The main features of three different types of data gathering constructions

	Type 1	**Type 2**	**Type 3**
Basic construction	SBC with sensor(s)	SBC master node and group of sensor nodes with sensors	Smart phone
Data gathering	No limitations – suitable for large data chunks such as photos	Suitable for low data transfer – SBC limitations	Device sensors – no hardware modifications
Data processing	SBC limitations	Mean value calculus, visualizing	Mean value calculus
Data storage	No limitations temporary storage	No limitations database storage	No limitations temporary storage

The division into two parts is enough for hardware when compared to the three types of data gathering devices in Table 1. On the hardware side, types 1 and 3 are embedded together - the sensors and processing capabilities are in one device. In type 2 the master node can control several sensor nodes.

The **sensor node** collects the data with sensors. The data could be simple data, such as temperature and humidity. On the other hand, the data could be more complex, such as photos. The hardware is selected according to the data requirement. The separation of sensor node is made because of type 2 where the idea is to use several sensor nodes with one master node. In type 2 construction sensors are connected to a single-board micro-controller, such as Arduino or similar, which can handle a lot of simple data. Simple data are numerical values. Type 2 sensor nodes are simple, low-cost devices. These are connected to the master node and the amount of transferred data should be in bytes or kilobytes. The preferred communication methods include Bluetooth, ZigBee, and LoRa for short distance, low rates, and low power consumption [25], [26]. Types 1 and 3 are similar to each other; both are physically one entity: Type 1 consists of sensors and an SBC such as Raspberry Pi. The collected data could be complex and may need processing power, for example, photos. Type 3 is smartphone based solutions. A typical smartphone has several sensors, e.g., gyroscope, accelerometer, and ambient light sensor [27]. For example, Android phones could handle a lot of simple data from their own sensors, or complex data such as photos from the phone's own camera.

The **master node** collects data from the sensor nodes. The master node has the capability to pre-process or temporarily store data if needed. The master node has a communication channel for a larger data transfer rate and long distance, for example, 3G / 4G / 5G, or WiFi are suitable. Depending on the master node's communication channel, remote control and configuration are possible. For example, with the Raspberry Pi the remote configuration is easily constructed with a suitable communication channel and Linux OS tools. The idea of a master node in type 3 smartphone based solutions is implemented with a self-made application, which handles data collection, storage, and processing. The application limitations come from the phone's OS and the fact that hardware changes or modifications have not been made.

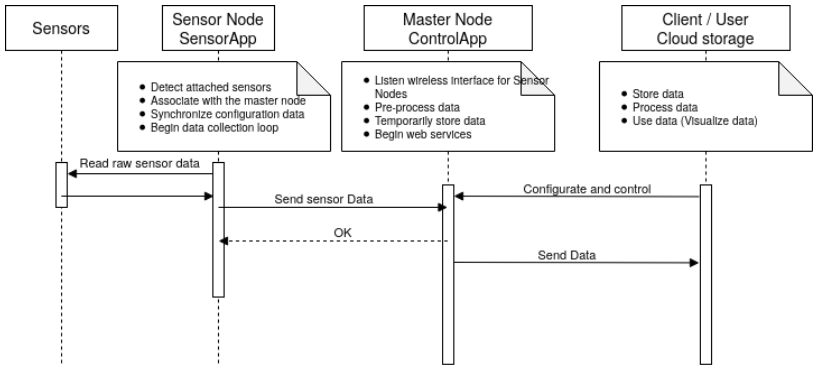

Fig. 7 Diagram of software component interconnections in the SW/HW framework.

This SW/HW framework relies on communication to the public Internet. The collected data is transferred via the Internet to the data storage devices. These could be cloud servers with a database or dedicated open source Linux servers for saving data. There are several database models for storing sensor data and each of these has a special use case where they are best. The SW/HW framework can be applied to all of these techniques.

4.2 Software of the SW/HW Framework

The hardware of the SW/HW framework also requires software. The software used is mostly open source. In this way the selected software is community tested and the source code is freely available. Open source software is also free to use. Therefore, several software combinations can be used for testing purposes without extra costs.

The software components are divided into three parts:

- Sensor software - gets sensor data from sensors
- Data gathering and pre-processing software
- Data storage software

Regarding how these three parts work together, the diagram in Fig. 7 is a guideline for dividing software components between the master node and sensor node on the abstract level. This kind of approach supports the modular development and interchangeability of components. The diagram also illustrates the input from the sensors and the output to the cloud.

The list of software components starts from the sensor node with sensor software and ends with data storage software for the master node. The sensing and sensor software typically have low-level programming with C++, Python, or Perl scripts, which are dedicated to do a few tasks, for example reading sensor data, reviewing the data, and storing the data. Sensor devices, such as Arduino with sensors, are

typically a micro-controller board that runs a dedicated program. Types 1 and 2 in Fig. 6 represent this kind of approach.

The Raspberry Pi based SBCs can perform both sensing and data gathering. In type 1 for example, the sensors could be connected to the Raspberry Pi and the sensor software reads the values from the sensors. If the construction is type 2, the gathering software handles the data collection from sensor nodes. The data gathering and preprocessing software are more complex and usually type 2 SBC devices are equipped with the full Linux operating system (OS). In the data storing and preprocessing phases, the SW/HW framework utilizes pre-made software and libraries. For example, Raspberry Pi could offer the gathered data to the Internet with a server application. Preprocessing in this scenario could be image recognition with image recognition software and library.

The gathered data are stored in the cloud server - this is the assumption of the SW/HW framework. Temporarily, the data could be saved to the master node using a suitable database. If the data meet the definition of a time series: "A sequence of numbers collected at regular intervals over a period of time" then a good choice is a time series database [28]. For example, the open-source time series database InfLuxDB is suitable for SBC hardware and is widely used in IoT solutions [29].

The other suitable database model for data storage is a relational model. The collected data could be stored locally in the sensor device, for example Raspberry Pi with Linux OS, MariaDB database, and a RESTful API combination. The RESTful API (Application Programming Interface) method allows remote control or management of a device over the network.

Smartphones equipped with the Android OS have been tested with this SW/HW framework. The Android OS has a software development kit (SDK), which enables the wide use of smartphone capabilities. For example, the SDK enables phone camera usage [23]. The SDK also enables usage of the smartphone's accelerometer sensor [30].

Data storage on the mobile phone is enabled by the OS. The SDK provides the capabilities to use files and databases for data storage. In terms of the SW/HW framework, the user should be able to use the data. The SDK also enables data transfer to cloud services.

Cloud storage for data is a better choice than local storage. Cloud storage could be, for example, a Linux server or maintained commercial cloud service such as Google Firebase. Both of these have more capabilities to store a larger amount of data than the local database in Raspberry Pi.

5 Validating the SW/HW Framework by prototyping

The SW/HW framework has been developed and tested during several research projects. The majority of IoT data gathering prototype systems and their findings have been reported in different studies [3], [31], [32], [30], [23], [22], [33]. The timeline of studies is presented in Fig. 8. The timeline also includes the first release of

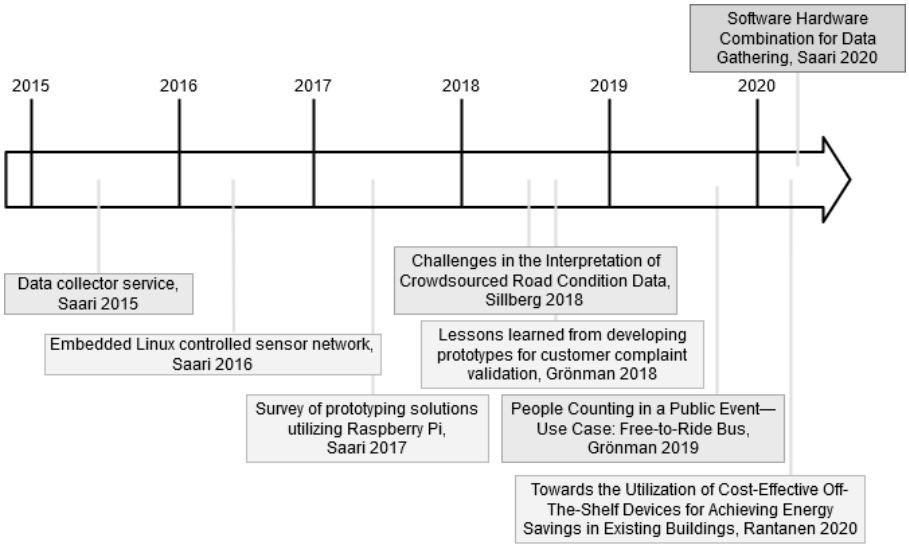

Fig. 8 Prototypes and validation in the SW/HW framework timeline.

the SW/HW framework study [34]. These projects have contained multiple iteration rounds. Each iteration has produced a working prototype system. This section gives an overview of the systems and lists the main findings during the development of the system. The systems are divided into the previously presented types 1, 2, and 3.

5.1 Type 1 and type 2 with SBC related prototype systems

Types 1 and 2 are SBC and Linux OS based data gathering prototype systems. Type 1 usually contains one sensor connected directly to the SBC.

- A data collector service [3] was the first implementation. The SBC was a Beaglebone Black which gathered temperature, humidity, and brightness data. The data were stored in the SBC and there was a service to offer the data to users. The SBCs were connected with an Ethernet connection. The main goal of the study was to experiment how well a cost-efficient SBC could be used to gather sensory data, and how this data could be provided to the client over the public Internet. This goal was reached successfully, and the designed system was tested and found to work as planned [3]. The study proved that fully working data gathering prototypes can be developed with off-the-shelf devices and open source tools.
- The ability to use off-the-shelf devices prompted us to find out how others have used these devices in academic research. A survey of prototyping was made to find out the benefits and limitations of Raspberry Pi [32]. Also, we searched for the testing methods of these prototypes. The study showed that the Raspberry Pi is a widely used device in research implementations of different kinds. Some

testing methods of prototype systems were found: software testing, software performance testing, and validation of data tests [32]. This study clarified the operating environment for the SW/HW framework.

- The third [23] and fourth [22] studies were similar to each other. Both prototypes presented in the studies were based on a Raspberry Pi and camera combination in a vehicular environment. The power supply was a battery backup and both used a 3G/4G communication channel. The data, photos, were transferred to the cloud storage where the data processing was handled. The focus of the studies was to analyze and process the data on a cloud server but the functionality of the SBC based prototype system was also ensured. Both of the systems were located in a client's environment for several weeks to collect data. The long testing time clearly showed that the SW/HW framework needs a configuration channel from user to device. In these systems, the SSH service and terminal were used as a remote channel.

5.2 Type 2 with SBC related prototype systems

The type 2 configuration has been tested in three real-world projects. The common construction of type 2 data gathering prototypes is that one master node SBC works with several sensor nodes. The communication is one-way from sensor nodes to the master nodes. The master node with a group of sensor nodes was tested in several research cases:

- The first was [31] where one SBC master node, an Intel Galileo Gen 2 development board, communicated using wireless XBee technology with several sensor nodes, on an Arduino development board. The collected data were environmental. The targets of the study were threefold: to test the model, to determine how well cost-efficient SBCs could be used to gather sensory data from several sensor nodes, and how to deliver this data to clients over the public Internet [31]. The study showed that SW/SW frameworks need a separate sensor node - master node architecture. This is useful when several sensors are required in a small area.
- The second use case [33] presented a wireless sensor system for monitoring indoor living or working conditions. The study expanded the range of sensors by using wireless LoRa technology in communication between the master node and sensor nodes. In the main construction, the master node Raspberry Pi received environmental data from several Sodaq sensor nodes. The data were stored in the master node for analysis and processing. The primary purpose of the tests was to validate the sensor system construction. Based on the experiments, we found that LoRa was a good choice for sensor applications within concrete buildings [33].
- The third use case used commercial, but open hardware and open source, RuuviTag sensor nodes [35]. These cost-efficient sensors collected temperature, humidity, pressure, and motion information. The data were sent using Bluetooth communication to the Raspberry Pi master node. The data were stored on an InfluxDB database and visualized with Grafana visualizing software. One of the

RuuviTag experiments is documented in a study [36]. This prototype system is quickly configurable, because RuuviTags do not need a configuration; only the Raspberry Pi needs a setup. These prototype systems showed that even though the gathered data were small in quantity, the Raspberry Pi limited the visualization. The second issue raised was a limitation in the amount of memory operations with a Raspberry Pi memory card. Therefore the prototype system experiment showed that data should be transferred and stored on a cloud server. From the perspective of the SW/HW framework, this prototype system showed the usability of modular development and the fact that the sensor nodes and master node are interchangeable.

5.3 Type 3: Smartphone related prototype systems

The smartphone is an excellent WSN sensor node. It has a working hardware package: power source, communication skills, and sensor devices. It also has a suitable OS, which allows the wide usage of the hardware. Our first ideas of using the smartphone as a sensor were presented in [37]. The main question presented was "How to utilize mobile technology to supply disaster information to both mobile terminals and desktop computers?" [37].

In the recent type 3 prototype systems, we used Android smartphones. These two documented data gathering prototypes are next discussed from the perspective of the SW/HW framework.

- The first data gathering prototype implementation was presented in study [30], which utilized data collected by smartphone sensors, such as an accelerometer and GPS, to detect variations in road surface conditions. The use of a smartphone was preferred because the data were obtained from a group of users driving on roads in western Finland and most of these people owned a smartphone that was good enough for prototyping purposes. The data gathering device was therefore an off-the-shelf Android smartphone without any hardware modifications. The software was a combination of user interface application and background service. The results of the study were that the basic programming task of creating a simple application for tracking the user's location and gathering data from the basic sensors embedded in a mobile device is a straightforward process. Minor difficulties arose because of variations in the phones' basic software and hardware depending on the manufacturer. In relation to the SW/HW framework, the prototype showed the ability to use a smartphone as a WSN sensor node.
- The second data gathering prototype was a solution for idea testing. The question asked was: Is it possible to take a photo if the phone is in a selected GPS position? Study [23] presents the working prototype solution for this problem. For the SW/HW framework, this prototype again showed the ability to use a smartphone as a WSN sensor node. The program code utilized an implementation of the haversine formula, which determines the great circle distance between two locations and is relatively simple to implement yet accurate enough for our use

cases. The application can be installed on any reasonably new Android device and takes advantage of the built-in sensors and camera of the device.[23]

6 Discussion

This study aims to resolve the research question: How to generalize the prototyping of IoT data gathering in a framework of required software and hardware components?

To answer the question, we made a reasonable number of data gathering prototypes and reported on them in academic research papers. Thus, our earlier research answered the question. The studies highlighted several aspects in constructing a SW/HW framework. The framework describes the main findings from developed prototypes. The framework brings out three different approaches for different use cases. The research papers presented how prototyping development can be made cost efficiently. This was enabled by using off-the-shelf embedded devices such as smartphones and Raspberry Pi SBCs. The devices have the type of OS that can be modified for sensor usage.

Furthermore, the prototype highlighted the knowledge we have about the process of prototyping. The model can be used as guidance when designing a new prototyping project together with a customer who wants to obtain information about some target environment.

The development of the SW/HW Framework raised several new ideas for research topics. These topics are briefly discussed here.

Each of the prototypes discussed has sensor software: software that reads the sensor, a temperature sensor for example, changes the value form bytes to an integer with a reasonable formula, and sends or stores the value somewhere. The study mentions the sensor software several times, but its construction has not been discussed in detail. This low-level program is coded using C/C++, Python, Java, or a similar programming language. It should be noted that at this level the initial data processing could be done, e.g., the mean value of accelerometer sensor values within one second. Is it possible to get improved performance without data loss? How should this low-level software be programmed? These would be suitable questions for future study.

The topics of user and user experience are beyond the scope of this study. Our prototypes were developed due to the needs of some project partner. The prototypes were tested with use case testing and once the customer had received a reasonable answer to a certain need, the development was stopped (except for one example [21] where there was a long piloting period in a real usage environment). The project outputs and prototypes are freely exploitable by the project partners.

The sensor prototypes produce a large amount of sensor data. Data processing and data mining are important issues, which this study leaves for future research, as it is such an extensive area. The issues of data visualization have been handled in some studies [30]. In addition, sensor data will become more usable if merged with

other publicly available data. This kind of data could be weather data or map data [38], [21].

Performance problems have not been extensively addressed in this study, but when using the SW/HW framework they have to be taken into account. In data gathering construction type 2, the amount of sensor nodes is limited, but no exact limit can be set. The limit is changed by the range, communication channel, data to be transferred and so on. The data processing can also affect the performance problems. For example, in this study photos are often mentioned as difficult regarding performance, especially motion detection. For example, photos should be transferred to a cloud server for processing.

The SW/HW framework does not set the quality criteria for components, but how can the selection of high-quality components be ensured? "The hardware quality depends on the price" is one claim, which in most situations makes sense. The second level for selection is "good enough". The SW/HW framework does not set these kinds of selection criteria for software or hardware components and therefore these decisions are left to the framework user.

The vulnerability of data is worth considering. Is the data critical and what happens if we cannot obtain the data? Is it possible to manipulate the data and what would the consequences be in that case? For example, what happens if somebody changes the data. This SW/HW framework does not take a position on the matter, but these are significant issues. Furthermore, security issues are important for IoT devices. Security vulnerabilities and attacks on IoT systems have been covered extensively by [39]. The SW/HW framework does not pay attention to security except for the communication channel. This concern was raised in [31] and the proposed, more secure, communication technology LoRa has been discussed by [40].

7 Summary

This paper introduced the software/hardware framework and a descriptive model for the prototyping process. The framework was developed during several research projects by following the same prototype development process. The model presented the process for constructing and testing a data gathering prototype with six steps, starting from discussion of requirements and ending with the presentation of collected data. The main aspects of these steps were presented briefly.

A sensor network consists of several layers, from data gathering devices to data users. The framework is placed in the data gathering layer. The three types of data gathering constructions were presented by introducing the software components, the hardware components, and their interconnections.

Research findings: The model and the framework were validated by presenting several previous research projects and studies.

Acknowledgements This work is part of the KIEMI ("Vähemmällä Enemmän – Kohti Kiinteistöjen Energiaminimiä", or "Less is More: Towards Energy Minimum of Properties" in English) project and has been funded by the European Regional Development Fund and the Regional Council of Satakunta.

References

1. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.
2. Ieee approved draft standard for an architectural framework for the internet of things (iot), 2019.
3. Mika Saari, Pekka Sillberg, Petri Rantanen, Jari Soini, and Haruka Fukai. Data collector service - practical approach with embedded linux. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, International convention on information and communication technology, electronics and microelectronics, pages 1037–1041. IEEE, 2015.
4. Mika Saari, Pekka Sillberg, Jere Grönman, Markku Kuusisto, Petri Rantanen, Hannu Jaakkola, and Jaak Henno. Reducing energy consumption with iot prototyping. *Acta Polytechnica Hungarica*, 16(9, SI):73–91, 2019.
5. Klaus Buchenrieder. Rapid prototyping of embedded hardware/software systems. In *Proceedings. Ninth International Workshop on Rapid System Prototyping (Cat. No.98TB100237)*, volume 5, pages 2–3. IEEE Comput. Soc, 2000.
6. C P Kruger, A M Abu-Mahfouz, and G P Hancke. Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 1926–1931, 2015.
7. Nick Babich. What is rapid prototyping?, 2019. Accessed: 2020-11-16.
8. Chris Jensen and Walt Scacchi. Discovering, modeling, and re-enacting open source software development processes: a case study. In *New Trends in Software Process Modeling*, number February 2006 in Series on Software Engineering and Knowledge Engineering, pages 1–20. World Scientific Publishing Company, 2006.
9. Ulrike Becker, Dirk Hamann, and Martin Verlage. Descriptive Modeling of Software Processes. *IESE-Report No. 047.97/E*, 1997.
10. Ulrike Becker-Kornstaedt and Richard Webby. A comprehensive schema Integrating Software Proces Modeling and Software Measurement. *IESE-Report No. 047.99/E*, 1999.
11. Edward Ashford Lee and Sanjit A. Seshia. *Introduction to Embedded Systems. A Cyber-Physical Systems Approach. Second Edition*, volume 195. 2017.
12. C. Kreiner, C. Steger, E. Teiniker, and R. Weiss. A HW/SW codesign framework based on distributed DSP virtual machines. *Proceedings - Euromicro Symposium on Digital Systems Design: Architectures, Methods and Tools, DSD 2001*, pages 212–219, 2001.
13. D. Saha, R.S. Mitra, and A. Basu. Hardware software partitioning using genetic algorithm. In *Proceedings Tenth International Conference on VLSI Design*, pages 155–160. IEEE Comput. Soc. Press, 1997.
14. M.B. Srivastava and R.W. Brodersen. Rapid-prototyping of hardware and software in a unified framework. In *1991 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*, pages 152–155. IEEE Comput. Soc. Press, 1991.
15. Li Da Xu, Wu He, and Shancang Li. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.
16. Alex Vakaloudis and Christian O'Leary. A framework for rapid integration of IoT Systems with industrial environments. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 601–605. IEEE, 2019.
17. IF Akyildiz, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

18. Mika Saari, Jari Soini, Jere Grönman, Petri Rantanen, Timo Mäkinen, and Pekka Sillberg. Modeling the software prototyping process in a research context. 2020. Accepted for publication.
19. Jere Grönman, Petri Rantanen, Mika Saari, Pekka Sillberg, and Juha Vihervaara. Low-cost ultrasound measurement system for accurate detection of container utilization rate. In *2018 41th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018.
20. Jari Soini, Pekka Sillberg, and Petri Rantanen. Prototype system for improving manually collected data quality. In Zoran Budimac and Tihana Galinac Grbac, editors, *Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2014, September 19-22, 2014, Lovran, Croatia*, Ceur workshop proceedings, pages 99–106. M. Jeusfeld c/o Redaktion Sun SITE, 2014.
21. Jari Soini, Markku Kuusisto, Petri Rantanen, Mika Saari, and Pekka Sillberg. A Study on an Evolution of a Data Collection System for Knowledge Representation. In A. Dahanayake, J. Huiskonen, and Y. Kiyoki, editors, *Information Modelling and Knowledge Bases XXXI*, volume 321, pages 161 – 174. IOS Press, 2019.
22. Jere Grönman, Pekka Sillberg, Petri Rantanen, and Mika Saari. People Counting in a Public Event—Use Case: Free-to-Ride Bus. In *2019 42th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2019.
23. Jere Grönman, Petri Rantanen, Mika Saari, Pekka Sillberg, and Hannu Jaakkola. Lessons learned from developing prototypes for customer complaint validation. In *Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, volume 2217, pages 27–30. CEUR Workshop Proceedings, 2018.
24. Fuewen Frank Liou. *Rapid Prototyping and Engineering Applications*. Taylor & Francis, CRC Press, Boca Raton, second edition, feb 2019.
25. Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications*, 23(5):60–67, oct 2016.
26. Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low Power Wide Area Networks : An Overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873, 2017.
27. Kodrat Iman Satoto, Eko Didik Widianto, and Sumardi. Environmental Health Monitoring with Smartphone Application. In *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pages 281–286. IEEE, 2018.
28. Dmitry Namiot. Time series databases. *Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL2015)*, 1536:132–137, 2015.
29. Andreas Bader, Oliver Kopp, and Michael Falkenthal. Survey and comparison of open source time series databases. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, 266:249–268, 2017.
30. Pekka Sillberg, Jere Gronman, Petri Rantanen, Mika Saari, and Markku Kuusisto. Challenges in the Interpretation of Crowdsourced Road Condition Data. In *2018 International Conference on Intelligent Systems (IS)*, pages 215–221. IEEE, 2018.
31. M. Saari, A. M. Baharudin, P. Sillberg, P. Rantanen, and J. Soini. Embedded Linux controlled sensor network. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1185–1189. IEEE, 2016.
32. Mika Saari, Ahmad Muzaffar bin Baharudin, and Sami Hyrynsalmi. Survey of prototyping solutions utilizing Raspberry Pi. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 991–994. IEEE, 2017.
33. *Towards the utilization of cost-effective off-the-shelf devices for achieving energy savings in existing buildings*. IEEE, 2020.
34. Mika Saari, Petri Rantanen, and Sami Hyrynsalmi. Software hardware combination for data gathering. In *Proceedings of 2020 IEEE 10th International Conference on Intelligent Systems (IS2020)*, 2020.
35. Ruuvitag technical specifications, https://ruuvi.com/files/ruuvitag-tech-spec-2019-7.pdf, 2019. Accessed: 2020-11-23.

36. Mika Saari, Jere Grönman, Jari Soini, Petri Rantanen, and Timo Mäkinen. Experimenting with Means to Store and Monitor IoT based Measurement Results for Energy Saving. In *2020 43th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2020.
37. Pekka Sillberg, Petri Rantanen, Mika Saari, Jari Leppäniemi, Jari Soini, and Hannu Jaakkola. Towards an IP-based alert message delivery system. In J Landgren and S Jul, editors, *ISCRAM 2009 - 6th International Conference on Information Systems for Crisis Response and Management: Boundary Spanning Initiatives and New Perspectives*, number June 2015, page 8 p. Information Systems for Crisis Response and Management, ISCRAM, 2009.
38. Pekka Sillberg, Mika Saari, Jere Grönman, Petri Rantanen, and Markku Kuusisto. Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data. In R Goncalves, V Sgurev, V Jotsov, and J Kacpzyk, editors, *Intelligent Systems: Theory, Research and Innovation in Applications*, Intelligent Systems: Theory, Research and Innovation in Applications, pages 99–119. Springer, 2020.
39. Sachin Babar, Antonietta Stango, Neeli Prasad, Jaydip Sen, and Ramjee Prasad. Proposed embedded security framework for Internet of Things (IoT). In *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, pages 1–5. IEEE, 2011.
40. M. Saari, A. Muzaffar bin Baharudin, P. Sillberg, S. Hyrynsalmi, and W. Yan. LoRa — A survey of recent research trends. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0872–0877. IEEE, 2018.

PUBLICATION VII

Modeling the Software Prototyping Process in a Research Context

Saari, M., Soini, J., Grönman, J., Rantanen, P., Mäkinen, T. and Sillberg, P.

Information Modelling and Knowledge Bases XXXII. ed. by Tropmann-Frick, M., Thalheim, B., Jaakkola, H., Kiyoki, Y. and Yoshida, N. 2020, 107–118
DOI: 10.3233/FAIA200823

Modeling the Software Prototyping Process in a Research Context

Mika SAARI, Jari SOINI, Jere GRÖNMAN, Petri RANTANEN, Timo MÄKINEN and Pekka SILLBERG
Tampere University, Faculty of Information Technology and Communication Sciences, Pori, Finland

Abstract. The paper examines the Third Mission of universities from the point of view of company collaboration in the prototype development process. The paper presents an implementation of university-enterprise collaboration in prototype development described by means of process modeling notation. In this article, the focus is on modeling the software prototyping process in a research context. This research paper introduces prototype development in a university environment. The prototypes are made in collaboration with companies, which offered real-world use cases. The prototype development process is introduced by a modeling procedure with four example prototype cases. The research method used is an eight-step process modeling approach. The goal was to find instances of activity, artifact, resource, and role. The results of modeling are presented using textual and graphical notation. This paper describes the data elicitation, where the process knowledge is collected using stickers-on-the-wall technique. Furthermore, the paper describes the creation of the model.

Keywords. Software process, process modeling, knowledge management, modeling methods, prototyping, modeling

1. Introduction

It is a common conception that the modern university serves three main purposes: teaching, research, and a broader social function. The latter of these functions, commonly dubbed "The Third Mission" [1-4], is considered to include measures contributing to social influencing and interaction. Nevertheless, multiple views in terms of defining the Third Mission exist, and Henry Mugabi [5] for example, compiled a selection of the varying definitions present in recent research literature in his dissertation. Moreover, the concept has been increasingly integrated into university strategies as well as operations pertaining to regional development [6]. Universities serve to produce and share knowledge and technological expertise, contributing in their part to the modernization and success of enterprises, and the "Third Mission" is often associated with tech-driven collaboration between the university and enterprises in addition to its social function.

The social significance of the Third Mission is widely acknowledged, and an international evaluation indicator conducted by UNESCO [7] places Finland among the top countries regarding collaboration between university and enterprises. In Finland, the most common manifestations of this kind of collaboration are various research and development projects that are often funded externally, examples of which include contributions by the Academy of Finland, Business Finland, various foundations, the

European Union, and other international sources. For small and medium-sized enterprises (SMEs) [8] in particular, collaborative projects with a university offer significantly better opportunities to participate in R&D activities for instance, as opposed to relying on internal funding and expertise only.

This paper introduces the model of prototype development practices that have been applied in research projects between university and enterprises (mostly SMEs) in Finland. The model has been used for many years in several collaborative projects between Tampere University, Pori unit, and regional enterprises with good success. In this paper, the model is presented at a detailed level with the help of a few cases serving as concrete examples. Promoting and contributing to the Third Mission of the university from the perspective of regional development, the format introduced in the present paper is but one example of collaboration between the university and enterprises.

The paper is structured as follows: In Section 2, background information along with work related to the subject are presented. The research approach in terms of the applied model is presented in Section 3. Section 4 introduces the process knowledge with the help of examples, after which Section 5 describes the creation of the Prototype Development Process (PDP) model in selected cases. Section 6 discusses the observations and challenges confronted during the use of the model. Finally, Section 7 summarizes the paper.

2. Related studies on the subject

This section describes some points of view, factors, and experiences, which relate to the collaboration context between universities and enterprises. Earlier research results in this field are examined below.

In their research, Basili et al. propose guidelines on how collaborative research could succeed between public sector research and industry. In [9], a couple of arguments are present, which are worthy of examination. First, they say that there is not enough research considering real development contexts. Second, they suggest that software engineering research needs to foster context-driven research if it seeks to evolve towards a brighter future. The main contextual factors are human, organizational, and domain-related. In [9], they believe that practical software engineering, the big picture, and suitable solutions are mostly born from bottom-up research and a succession of case studies rather than from top-down research.

In context-driven research it is necessary to have intense collaboration between industry and academia. There need to be cycles of model building, experimentation, and learning in software engineering research. Usually researchers face some problems in identifying the challenges of collaborating organizations. After overcoming these challenges, the gap between the state of the art and the requirements for a solution must be assessed. In developing a solution, it is important to clearly define working assumptions in order to achieve applicability and scalability in context [9].

Another major part of this collaboration relates to funding. There are a couple of ways to start funding a research project between industry and academia. Companies can give grants to students/researchers for a specific project, build collaborative projects with academia in short- or long-term existing relationships, and there is an opportunity to build up a wider network of partners between industry and academia [10]. Also, governments are encouraging this kind of collaboration because it improves innovation

efficiency and thereby enhances wealth creation [11]. As a result, several countries have already put innovative programs in place [9].

Industry-academia collaboration benefits those organizations that do not have their own R&D facilities. Companies can utilize the knowledge of academic resources to understand their modern-day software engineering problems. Industry has noted that it can support innovation and development processes when collaborating with researchers [9].

Companies are increasingly investing in software development, although their core areas of expertise are defined around business areas and systems rather than software. However, most companies do not have the necessary resources and know-how to develop effective solutions to software engineering problems. This makes collaboration between industry and academia very useful [9].

The aim of a study by Wohlin [12] was to gather experiences and lessons learned from successful collaboration between industry and academia in two different environments. First, it was performed in Sweden and included a six-year collaborative project. The collaboration partners were five different-sized companies from various sectors and the Blekinge Institute of Technology. Later the study was replicated in Australia. Industry roles included product managers, project managers, developers, and testers. Academic roles included professors, researchers, and students at different levels.

The results from Wohlin's study [12] were that support from company management is crucial. There must be a champion at the company who argues for the cause, and not only a person assigned the responsibility for the project. There are different levels of understanding between different categories of people (for example, people in industry, senior researchers, and students). Social skills are particularly important in long-term collaboration.

In the study by Carver [10], there is an example of industry-academia collaboration. The background of the study was that the challenges faced by the companies were too labor intensive, lacked context-specific validation, or were not embedded into existing tools or design processes. Collaboration between industry and academia produces successful results when there is a good connection between academic and industry partners, there are the right collaborators on both sides, the timing of the interaction fits the requirements on both sides, and it is understood that the process from research prototypes and an academic publication to a deployed solution takes time.

In [13], the author interviewed researchers to understand their needs and problems in research-based projects. University research was more concerned with community issues, and companies had clients that were constrained by formal agreements. The industry groups had defined roles and responsibilities; the research teams were more dynamic. Industrial companies used formal development methods, but most university projects did not. The milestones for industrial projects were set by the customer, while the milestones for university projects were usually set by the funding partners. After the interviews, the author decided, for a number of reasons, including the uncertainty about the research objectives, that university researchers were unable to use a well-defined software development method.

The above-described examples provide the research background for the context that is discussed in this paper. The following section concentrates on explaining the basis of the process modeling, which is relevant background information for the description later, in Section 4, of the implementation of the university-enterprise collaboration with the help of the process modeling notation.

3. Research approach – An eight-step model

This section deals with software process modeling in a research context. Modeling is an approach for analyzing and understanding a complex phenomenon resulting in a model, which is a simple and familiar structure that can be used to interpret some part of reality [14]. When the phenomenon to be analyzed is a software process, information is captured and classified into a model with the help of a process-modeling schema [15], i.e., a meta-model specifying the concepts, relationships, and rules [16] used when modeling processes. The basic concepts related to the software process include activity, artifact, resource, and role [17].

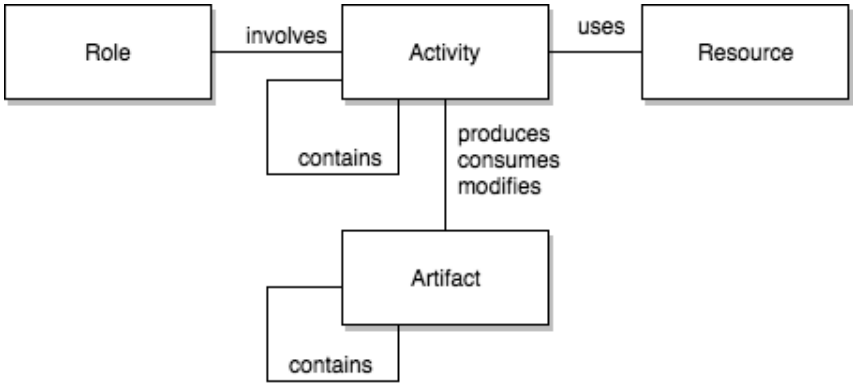

Figure 1. The basic concepts related to software processes.

The results of modeling are presented using a textual or graphical notation. There are several approaches for eliciting information for process models [15] such as interviews and artifact analyses. Process modeling can be prescriptive or descriptive. While a Prescriptive Process Model (PPM) describes how a process should be performed, a Descriptive Process Model (DPM) describes how it actually is performed [18].

In this study our aim is to model software development practices performed in an academic context. We follow the DPM approach proposed by Becker & al. [19]. The approach consists of eight steps grouped into two phases:

Set-up phase

1. Objectives and Scope
2. Define Schema
3. Select Language
4. Select and Tailor Tools

Execution phase

5. Elicitation
6. Create Model
7. Check Model

8. Check Process

We will apply the DPM approach in the following way: The data for the models is collected through interviews with the developers involved in the processes. The schema shown in Figure 1 guides the data collection, the results of which are shown on stickers on the wall during the work (Figure 2 in Section 4). In the resulting models, the activities are represented as rectangles with rounded corners. Stick figures represent roles and different icons represent resources. Artifacts appear as parallelograms, cylinders, and document symbols. The associations between activities and artifacts are represented by continuous arrows and the links between activities and roles and resources are dashed. Gray symbols and dashed rectangles represent aggregations (Figures 3-6 in Section 4). Graphical representations of the models are produced by a free online diagram software, draw.io. (DPM Steps 2-4).

The objectives and scope of the modeling are presented in the introduction of this paper (Step 1). The following section provides an example of data elicitation (Step 5) including the modeling results (Step 6). The last section of the paper discusses the possibilities of improving the modeled processes (Steps 7-8).

4. Process elicitation and resulting models

This section describes the steps 5 and 6 of the DPM approach. Process knowledge is highlighted in this section, which introduces four different prototype development processes (PDP 1-4).

Information for the process models is collected from four cases:

PDP 1 - Verification of customer complaints related to bus routes [20].
PDP 2 - Verification of customer complaints related to garbage collection. [20].
PDP 3 - Data collection in a public indoor swimming pool [21, 22].
PDP 4 - Passenger counting in a free-to-ride bus [23].

Common to all of these example cases is the software development resulting in a working prototype. The development process starts from the idea of collecting certain data with certain equipment. Then the idea is validated – can it be viably implemented? If the answer is yes, the implementation phase produces the first working prototype. Usually prototype implementation includes software coding and the implementation of hardware from off-the-shelf devices. The working prototype is tested in a laboratory and if the device displays sufficient reliability, the device is moved to real-world testing. The development processes and testing phases usually produce data. The overall outcome from these prototype development processes has been academic output such as a research paper. Knowledge of the prototyping process is visualized as illustrated in Figure 2.

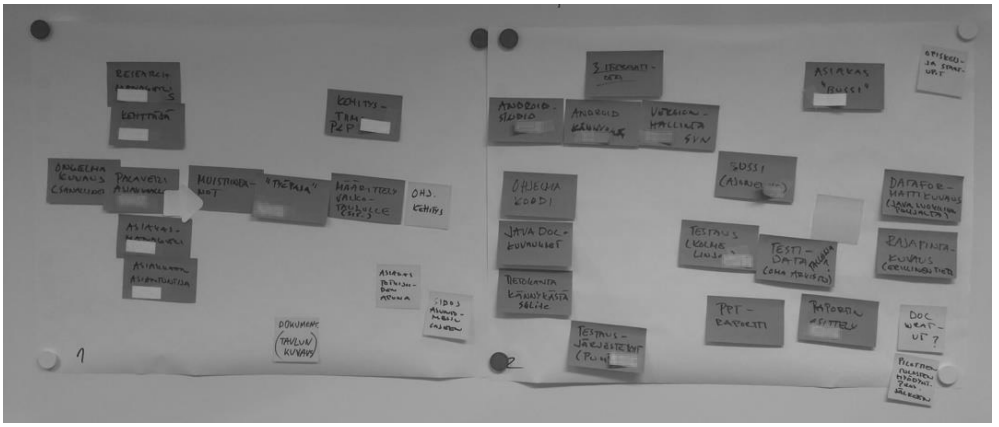

Figure 2. Whiteboard and notes.

This whiteboard and the Post-It notes are one way to highlight development process practices [24]. The notes are color-coded: Yellow markers are roles, green denotes activity, red is for resources, and the blue notes are artifacts. The orange notes describe issues and improvement ideas that came up during the data elicitation process. Knowledge of the development process is collected by means of this whiteboard and note notation (stickers-on-the-wall technique).

Further, this section presents the creation of a model for each of the PDPs. First we introduce the developed model for PDP 1 and 2. Figure 3 presents the university – company interaction. The model includes six steps which have been identified from the development process. The steps start from requirements definition and end with the publication of results. The gray boxes – the software development step and the development artifacts – are discussed more in the subsections on the bus case and the garbage collection case.

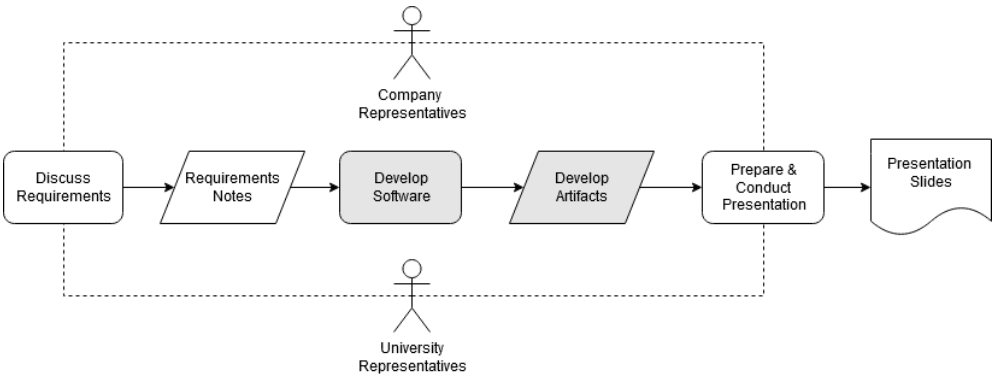

Figure 3. The illustration of university – company interaction during development (common to all PDPs).

The model includes all the main factors. The university representatives are the research group including the project manager and software/hardware developers. This

group has the main responsibility for the prototype development. The company representatives are involved in the development process in the role of instructor. In the presented PDPs 1-4, the company representatives are active at the starting and ending points: at the start with the definition of requirements and at the end where the results are presented to them. They also provide the testing environment if the testing is done at their company. However, they are not involved in the development process itself.

4.1. Verification of customer complaints related to bus routes (PDP 1)

The bus case (PDP 1) was established for handling customer complaints. The bus company had received complaints from customers that the buses did not stop to pick up customers or did not come at all. The prototype was developed to collect photos with time stamps at defined bus stops. This prototype was implemented in mobile phones and the main idea of prototype was to collect photos of the bus stops as the bus approached. The project group at the university developed a working prototype, which was then tested in the buses [20].

The development process is illustrated in Figure 4, expanding the previously presented steps "Develop Software" and "Develop artifacts". The development group in this process consists of only university personnel. The development process starts with a design discussion – the first activity, which produces the first artifact: the whiteboard sketch. The results are then used in the software construction phase. This activity produces the second artifact: software code with documentation. After that the test activity starts, which produces the collected data artifact. The coding and testing activities could be iterated several times. Figure 4 also includes the resources used during the development process: programming language, test device with GPS, camera and network, and the testing environment – the bus itself.

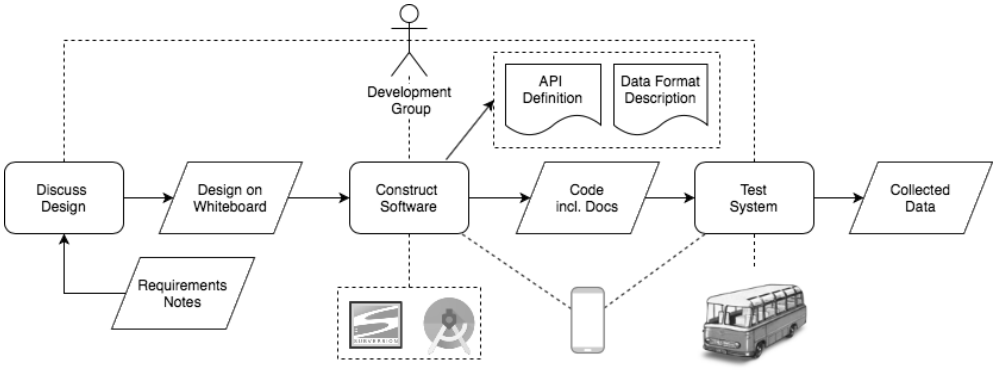

Figure 4. The development process of the bus case.

The presented development process produces a working prototype. In this case, the prototype was introduced to the customer – the bus company.

4.2. Verification of customer complaints related to garbage collection (PDP 2)

The garbage collection case (PDP 2) was similar to PDP 1. The use case was also intended for managing customer complaints. The garbage collection company had received complaints from their customers that the trashcans had not been emptied. In most cases the reason was a vehicle blocking the garbage collection truck or similarly that the truck was unable to empty the trashcan. The prototype solution in this case used the same idea as in the bus case: namely, put the camera in the truck and take pictures when the GPS registers the right location [20].

Figure 5 shows the activities and artifacts. The biggest difference was in the resources: the cellphone was replaced by a Raspberry Pi with a camera, GPS, and network device. The data collector resource was the MySQL server instead of the phone. In addition, the testing environment was the garbage collection truck itself.

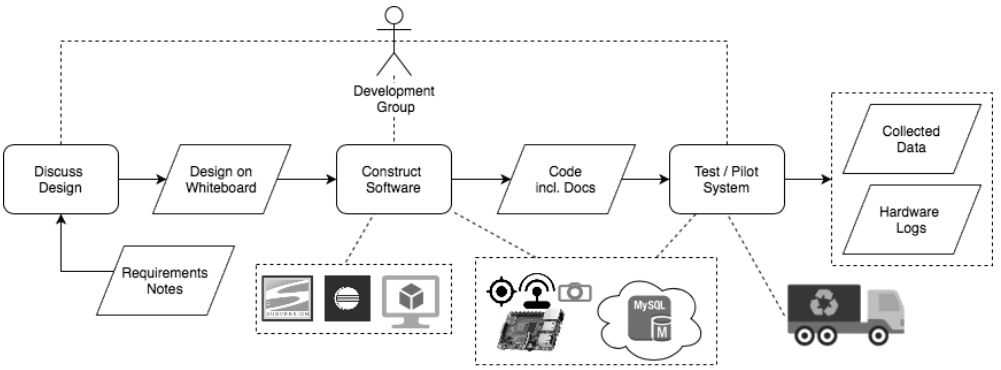

Figure 5. The development process of garbage collection case.

The developed prototype worked and the piloting phase in the garbage truck lasted several weeks. The development process captured data, which in this case were photos. In addition, the device – a Raspberry Pi, generated a test log during the pilot phase.

4.3. Combined model of bus and garbage collection cases

PDP 1 and PDP 2 should be presented together because the second prototype – the garbage truck camera system – utilized the definitions and results of the first prototype. They were also implemented very close to each other in time.

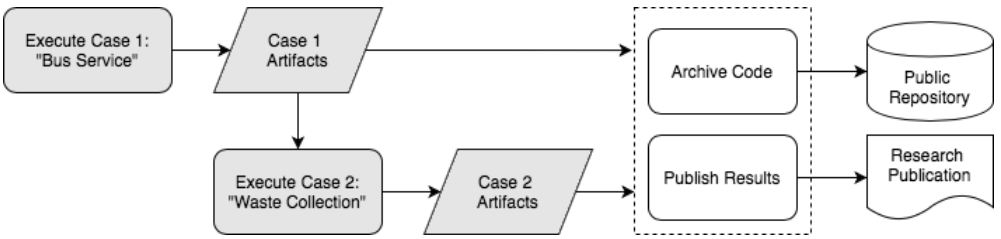

Figure 6. The combined model of the bus and garbage collection cases.

The outcomes of these PDPs were a public repository and research publication [20]. The public repository includes all the developed software code.

4.4. Data collection in a public indoor swimming pool (PDP 3)

PDP 3 handled a prototype system with the purpose of improving the quality of manually collected data. The prototype was a mobile application that the maintenance personnel used to collect and store data from several different meters in a public swimming pool. More information on the prototype can be found in the research articles by Soini [21, 22].

The Prototype Development Process was similar to the one shown in Figure 3. Activities included the discussion of requirements, software coding, and the presentation of the results. In addition, the artifacts were similar: discussion notes, software, and a research publication. The biggest difference to the other cases was that the implemented prototype remained in use after the pilot phase. This "extended piloting" period is handled in the research paper by Soini [22] along with aspects of software evaluation. Also, long-term piloting is examined from the point of view of system developers, administrators (maintenance), and end users [22].

4.5. Passenger counting in a free-to-ride bus (PDP 4)

PDP 4 handled a prototype system with the aim of counting passengers on a free-to-ride shuttle bus. Unlike an ordinary people counter, the customer wanted more information on where and when passengers got on and off [23].

The Prototype Development Process was similar to the one shown in Figure 3. Activities included the discussion of requirements, software coding, and the presentation of the results. Again, the artifacts were similar: discussion notes, software, and a research publication. The difference compared to PDP 1-3 was that this prototype was developed for a real-life use case of collecting statistics about bus passengers on a free-to-ride shuttle bus route at a large public event in the summer of 2018 in Pori, Finland. In this case, the development process ended in the one-month pilot. The outcome of the pilot was the presentation of the pilot results to the customer.

5. Discussion

The process introduced and modeled in this paper has been used to create multiple prototypes and pilot experiments during recent years. Thus, based on the observed results, it can be considered to be successful and fit the needs of our development cases. However, while discussing past projects with the team members several challenges did come up. Furthermore, while collecting data for the model, notes were made of issues that the team members pointed out (the orange notes on the whiteboard in Figure 2).

The first issue was the documentation of intermediate specifications described on whiteboards. The funding or goals of the projects do not especially require extensive intermediate documentation, and in practice, only very rarely has there been the need to study the intermediate specifications created during the process. The low requirements

for documentation have probably been one of the main reasons for the bad habits in documentation practices. Generally, the decision of whether to prepare any documentation has been based on the developer teams' "gut feeling" about how complex the specification was. In other words, "proper" documentation has been created for more complex intermediate specifications, but simpler specifications and drawings have not been documented in any way. Nowadays, it is quite simple to use a smartphone to capture the information on whiteboards, so in the future, it might be better to document everything systematically.

The second issue was the interaction with the customers (e.g., companies) - or the lack of it - during the actual prototype development process. The interaction has often been limited to the use case definition phase, to the organization of practical arrangements of pilots (e.g., agreements on which bus lines could be used for testing prototypes), and to the presentation of the research findings. In the final presentation meetings, the companies have never indicated that they would have liked to be more involved in the process. The feedback from the companies has mainly been related to the research findings, and the developed prototypes, and not to the development process itself. In our case, the companies have often not been software-oriented, which could have had an effect on their interest in the process, and it could also have limited the advantages achievable by involving them further in the process. In addition, the companies did not (directly) invest any resources (money or personnel) in the projects. This might have further reduced their interest in participating more deeply in the development process. Furthermore, as the companies had their actual business to run, there could have been challenges in creating a common schedule for meetings for all the parties involved. Of course, depending on the outcome of the research and pilots the companies can obtain knowledge, business ideas, or even working software to use in their actual business, but during the development these results may be too abstract to evoke deeper interest. In addition, the fact that the issue was not brought up in the meetings does not necessarily mean that there is a lack of interest in deeper involvement from the company side. As the university team did not especially raise the issue, it might be that the companies felt that they simply did not want to interfere in the university practices. Thus, to improve our model, the actual company interest in the prototype development process should be further studied.

Third, the subject of the usability of the project results came up. In principle, anyone can use the results because the codes and documentation are published as open source, but no studies have been performed on how or if the results are actually used. In general, after the projects (and thus, funding) has ended, the results have been left "as is" without maintenance, bug fixes, or feature improvements. The purpose of the projects was not to create "end products", and usually productization is not one of the project goals, leaving the created software and hardware applications in a state that would require further development into an end-user quality product. Also, it can be slightly challenging to find the material from, for example, the GitHub repository if one does not know exactly what one is looking for. Internally, the published codes and specifications have been reused in future projects when applicable. One potential future direction for research would be a study on how the results of university projects should be published to be most useful for outside parties, and what the crucial elements are that should be published — or are the elements practically the same as in any other prototype development project?

Finally, the participation of university students was discussed. The advantage of involving the students more deeply would be to give the students more meaningful task

assignments (for example, for programming or other software engineering courses), but in practice, in the past the participation of students has been rare, and has mainly been limited to PhD students who have been hired by the university or worked at the university on their own funding. In the future, the model introduced in this paper could also be expanded to describe the involvement of students.

6. Summary

The paper examined the Third Mission of universities from the point of view of company collaboration in the prototype development process. The paper presented an implementation of university-enterprise collaboration in prototype development described by means of process modeling notation. The process introduced and modeled in this paper has been used to create multiple prototypes and pilot experiments over recent years. Thus, based on the observed results it can be considered to be successful and fit the needs of our development cases. The prototypes were made in collaboration with companies, which offered real-world application problems. The prototype development process were introduced by a modeling procedure with four example prototype cases. The research method used and presented here was an eight-step process modeling schema. The basic concepts relating to the software process included four factors: activity, artifact, resource, and role. The results of the modeling were presented using textual and graphical notation. This paper described the elicitation of process knowledge. Furthermore, the paper described the creation of a model. The PDP model provides one concrete and systematic example of how university-enterprise collaboration can be executed in practice. Moreover, the model presented is a real-life indication of how the Third Mission task set for universities can be successfully implemented.

References

[1] F. Schutte, and P. C. van der Sijde, “The University and its region”. Examples of regional development from the European Consortium of Innovative Universities. Twente University Press, Enschede, Netherlands. 2000.

[2] T. Vorley, and J. Nelles, “Building Entrepreneurial Architectures: a conceptual interpretation of the Third Mission”, Policy Futures in Education, Vol. 7, No. 3, pp. 284-296, 2009.

[3] I. Niiniluoto, “Yliopistot ja ammattikorkeakoulut yhteiskunnallisina vaikuttajina”, In the publication Vastuullinen ja vaikuttava. Opetus- ja kulttuuriministeriön julkaisuja 2015:3, pp. 11-30, 2015.

[4] K. Kankaala, E. Kaukonen, P. Kutinlahti, T. Lemola and M. Nieminen (2004), Yliopistojen kolmas tehtävä ?, Edita Publishing Oy. Helsinki, pp.15-42, 2004.

[5] H. Mugabi, Institutionalisation of the ”Third Mission” of the University. The case of Makerere University. Academic Dissertation, Tampere University Press, Finland 2014.

[6] A. Zomer, and P. Benneworth, “The Rise of the University’s Third Mission”, Reform of Higher Education in Europe, (eds.) Enders, J., de Boer H.F. ja Westerheijden, D.F., Sense Publishers. Netherlands, pp. 81-102, 2011.

[7] UNESCO (2015), UNESCO science report: Towards 2030, p. 74, https://unesdoc.unesco.org/ark:/48223/pf0000235406/PDF/235406eng.pdf.multi (Retrieved 4/1/2020 World Wide Web)

[8] M. Marchesnay, “Fifty years of entrepreneurship and SME: A personal view”. Journal of Small Business and Enterprise Development, Vol. 18, No. 2, pp. 352-365, 2011.

[9] V. Basili, L. Briand, D. Bianculli, S. Nejati, F. Pastore and M. Sabetzadeh, "Software Engineering Research and Industry: A Symbiotic Relationship to Foster Impact," in IEEE Software, Vol. 35, No. 5, pp. 44-49, 2018.

[10] J. C. Carver and R. Prikladnicki, "Industry–Academia Collaboration in Software Engineering," in IEEE Software, Vol. 35, No. 5, pp. 120-124, 2018.
[11] T. Barnes, I. Pashby, A. Gibbons, "Effective University – Industry Interaction: A Multi-case Evaluation of Collaborative R&D Projects," in European Management Journal, Vol. 20, Issue 3, pp. 272-285, 2002.
[12] C. Wohlin, A. Aurum, L. Angelis, L. Phillips, Y. Dittrich, T. Goschek, H. Grahn, K. Henningsson, S.Kagstrom, G. Low, P. Rovegard, P. Tomaszewski, C. van Toorn and J. Winter, "The Success Factors Powering Industry-Academia Collaboration," in IEEE Software, Vol. 29, No. 2, pp. 67-73, 2012.
[13] D. M. P. Dias, "Managing research based software product development in Sri Lankan universities", University of Colombo School of Computing, 2016.
[14] M. Boman, J. Bubenko, P. Johannesson and B. Wangler, "Conceptual Modeling". Prentice Hall, 1997.
[15] U. Becker-Kornstaedt, "Prospect: a Method for Systematic Elicitation of Software Processes". PhD dissertation. Technische Universität Kaiserslautern, 2004.
[16] ISO/IEC 24744:2006. Software Engineering — Metamodel for Development Methodologies. Draft International Standard ISO/IEC FDIS 24744:2006(E), ISO/IEC JTC1/SC7, Montréal, Québec, Canada.
[17] U. Becker-Kornstaedt and R. Webby, "A Comprehensive Schema Integrating Software Process Modeling and Software Measurement". Research Report 047.99/E, Fraunhofer IESE, Kaiserslautern, Germany, 1999.
[18] C. Jensen and W. Scacchi, "Discovering, Modeling, and Re-enacting Open Source Software Development Processes: A Case Study". In Silvia T Acuña, S. T. and Sánchez-Segura, M. I. (eds.). New Trends in Software Process Modeling. World Scientific Pub. Co., 2006.
[19] U. Becker, D. Hamann, D and M. Verlage, "Descriptive Modeling of Software Process". Research Report ESE-Report, 045.97/E, Fraunhofer IESE, Kaiserslautern, Germany, 1997.
[20] J. Grönman, P. Rantanen, M. Saari, P. Sillberg, and H. Jaakkola, "Lessons Learned from Developing Prototypes for Customer Complaint Validation", Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Novi Sad, Serbia, August 27-30, 2018.
[21] J. Soini, P. Sillberg, and P. Rantanen, "Prototype System for Improving Manually Collected Data Quality," in Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2014, Lovran, Croatia, pp. 99–106, 2014.
[22] J. Soini, M. Kuusisto, P. Rantanen, M. Saari and P. Sillberg, "A Study on an Evolution of a Data Collection System for Knowledge Representation", EJC 2019: Proceedings of the 29th International Conference on Information Modelling and Knowledge Bases, 2019.
[23] J. Grönman, P. Sillberg, P. Rantanen and M. Saari, "People Counting in a Public Event—Use Case: Free-to-Ride Bus", 2019 42th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019.
[24] A. Raninen, J. J. Ahonen, H.-M. Sihvonen, P. Savolainen, and S. Beecham, "LAPPI: A light-weight technique to practical process modeling and improvement target identification," Journal of Software Evolution and Process, Vol. 25, No. 9, pp. 915-933, Sep. 2013.

PUBLICATION VIII

Lessons Learned from Collaborative Prototype Development Between University and Enterprises

Harjamäki, J., Saari, M., Nurminen, M., Rantanen, P., Soini, J. and Hästbacka, D.

Proceedings of the 33th International Conference on Information Modelling and Knowledge Bases. Ed. by Welzer Družovec, T., Hölbl, M., Nemec Zlatolas, L. and Kuhar, S. 2023, 273–300

DOI: https://doi.org/10.18690/um.feri.5.2023.13

Lessons Learned from Collaborative Prototype Development Between University and Enterprises

Janne HARJAMÄKI [a,1], Mika SAARI [a], Mikko NURMINEN [a], Petri RANTANEN [a], Jari SOINI [a] and David HÄSTBACKA [a]

[a] *Tampere University*

ORCiD ID: Janne Harjamäki https://orcid.org/0000-0002-4595-7231, Mika Saari https://orcid.org/0000-0001-7677-2355, Mikko Nurminen https://orcid.org/0000-0001-7609-8348, David Hästbacka https://orcid.org/0000-0001-8442-1248

Abstract. In this article, the focus is on the KIEMI research project ("Less is More: Towards the Energy Minimum of Properties" in English) conducted in Tampere University during the period of 2019-2022. In this project, we used the earlier developed Descriptive Model of Prototyping Process (DMPP) to guide university-enterprise collaboration. The project consisted of several pilot cases, with prototypes, which were done in collaboration with companies, tackling real-world problems. In this article, we review and evaluate the suitability of the DMPP for usage in a research project. The article explores the topic from two directions: the collaboration of university and enterprises, and the reusability of artifacts within the DMPP. The paper introduces several pilot cases made on the KIEMI project, and describes the usage of the DMPP in them. Furthermore, the paper evaluates the model, sets forward the challenges faced, and, finally, discusses topics for future research.

Keywords. Artifact, Reusability, Collaboration, DMPP

1. Introduction

Universities and other research organizations produce research results, typically in the form of publications, such as papers and technical reports. In addition, applied research produces prototypes with proofs of concept (PoC). This study presents the outcome of one university project, where proofs of concept were mainly implemented by building data-gathering prototypes.

The focus of this study is on the findings of the KIEMI project ("Vähemmällä Enemmän – Kohti Kiinteistöjen Energiaminimiä", or "Less is More: Towards the Energy Minimum of Properties" in English). The aim of the project was to develop proof-of-concept demonstrations and prototype applications that illustrate how cost-effective, open, and modular solutions could be utilized to improve the energy efficiency of ex-

[1] Corresponding Author: Mika Saari, mika.saari@tuni.fi

isting, older buildings [1]. The KIEMI project was selected for analysis in this paper because of its large number of pilot use cases.

The goal of the KIEMI project was to save energy, and we worked towards this goal by developing and constructing data-gathering IoT sensor systems. We used the developed SW/HW framework [2] and the formerly developed descriptive model of the prototyping process (DMPP) [3]. The SW/HW framework generalizes prototype development into a group of necessary components and even more precisely the framework defines guidelines for constructing prototype systems to collect data for different purposes by reusing the required software and hardware components [2]. The DMPP was developed to guide the IoT prototype development process and can be used as a guideline when building a prototype. The DMPP contains the prototype development practices that have been applied in research projects between our university and enterprises. With these developed IoT prototypes, developers can receive valuable feedback on the possibility of implementing the application [3].

The following research questions were formulated during the project work. For this study, we wished to gain insight on the following topics:

- RQ1: Collaboration. How was university-enterprise collaboration executed in practice using the DMPP?
- RQ2: Reusability. How did the reusability of the artifacts in the DMPP steps support the workflow of the pilot cases?

University-enterprise collaboration (part of universities' third mission [4], [5]) has been used in previous projects and the DMPP model was developed into its current format based on the pilot cases of these previous projects. The KIEMI project also aimed to build prototypes in collaboration with companies for IoT type data gathering. Since we already had a completed process template, it was decided to put it to good use in this project as well, and RQ1 looks at the success of this issue.

Further, RQ2 focuses on the operation of DMPP sub-processes and how templates were created from them. The use of templates was intended to accelerate the operation. At the beginning, their significance was not understood, but by following the model the usefulness of the templates was noted. The same practices were observed when using the process model, so reuse was included in the review. The benefit and reusability of templates created specifically from reporting was monitored as it was expected to speed up the implementation of some steps.

The structure of this paper is as follows: In Section II, we review the related research about universities' third mission, industry collaboration. Also the background of the KIEMI project is explained. In Section III, we introduce the DMPP and its connections with project work. Further, the implementation of university-enterprise collaboration in prototype development is described by means of process modeling notation. Section IV introduces the KIEMI project – its purpose, activities, goals, and outcome. Section V continues by describing the prototyping pilot cases performed during the KIEMI project. Section VI evaluates the usability of DMPP in the KIEMI project highlighting results of the project and pilot cases. Section VII summarizes the study, and includes a discussion and suggestions for future research on the topic.

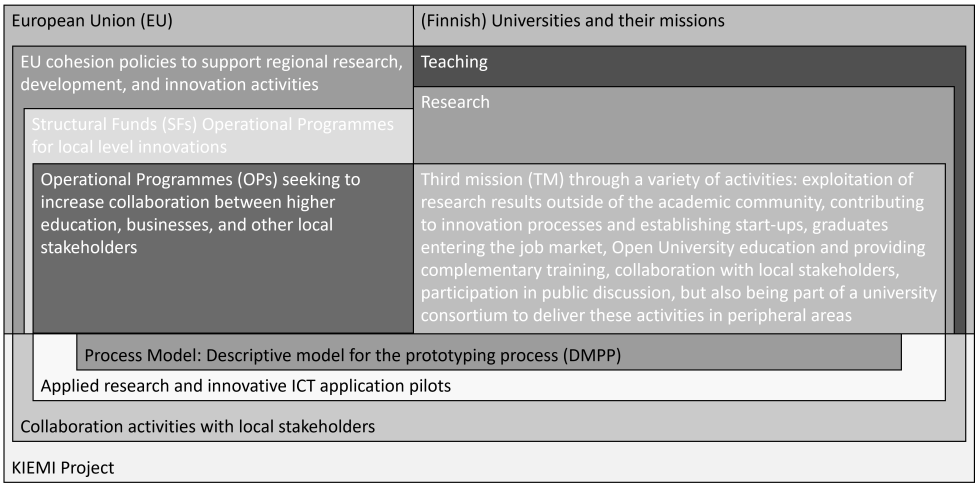

Figure 1. Third mission concept with the KIEMI project

2. Background

2.1. Third mission

It is a common conception that the modern university serves three main purposes: teaching, research, as well as a broader social function. The latter of these functions, commonly dubbed ”The Third Mission” [4], [5], is regarded as including measures contributing to social impacts and interaction.

Industry-academia collaboration benefits those organizations that do not have their own R&D facilities. For example, companies can utilize the resources of a university to understand their modern-day software engineering problems. Industry has realized that it can support innovation and development processes when collaborating with researchers. [6]

Figure 1 illustrates how the process model approach can be used to align European Union policy and Finnish universities' missions in the form of applied research and collaboration.

The EU cohesion policy and EU Structural Funds (SF) are used through Operational Programmes (OPs) to make it possible to create innovative collaboration projects for local stakeholders. Finnish universities have extended their traditional teaching and research activities within the third mission (TM) to exploit research results for peripheral areas, i.e., in the form of collaboration with local stakeholders. [7]

The University Consortium of Pori (UC Pori) has longstanding and specialized experience of creating collaboration with local stakeholders using the EU SF and OPs through university facilities and resources [7]. The KIEMI project represents a continuation of the series of OPs executed at UC Pori in recent years.

In collaboration, the transfer of technology is an important part, because it innovates development processes and innovative products achieve improved business competitiveness. In the study by [8], innovation is considered as a process consisting of two phases: technology creation and technology transfer.

As seen in Figure 1, the KIEMI project was a framework for implementing collaboration and applied research methods in the form of innovative ICT application pilot cases for local stakeholders. The descriptive model for the prototyping process (DMPP) was the spearhead of the process, pulling all the pieces together.

2.2. Collaboration channels for interactions

Interaction between public research organizations and industry can be implemented through many kinds of collaboration channels. One way to classify collaboration channel types was done in [9], where channels were divided into four groups: traditional, services, commercial, and bi-directional. In this paper, collaboration in SF OPs can be seen as bi-directional collaboration between university and industry, where both parties benefit from the acquisition and development of the technological know-how necessary for the prototype. In addition to the technical content, the prototype usage must take into account the development of interconnections necessary for university-enterprise collaboration and their impact on future cooperation activities.

2.3. Innovation models for collaboration

In projects like KIEMI, collaboration activities are done several times; mostly each time with different SMEs or public organizations (or some unit or department from their organization). To simplify this for the reader, we use the term industrial development (ID) for these collaboration parties or stakeholders. In addition, in case some ID has their own research group or department or if there is a CEO with a researcher's mindset, their staff can be referred to as industrial research (IR). Similarly, the university research unit, as in the KIEMI project, can be defined as academic research (AR).

For successful collaboration management between ID and AR, it is useful to have a framework or process model to ensure that the collaboration and innovation activities inside it create solutions and PoC along with pilot cases and receive strong support from all parties from the very beginning.

In the study by Punter[8], two main stakeholder groups were identified: researchers and industrial practitioners, where the former (AR) act as a technology provider and the latter (ID) as a technology receiver. They also pointed out that AR and ID may have completely different values and targets for technology and collaboration activities. AR is interested in proving concepts for technology via pilot cases during projects. ID is looking for a statement or evaluation of the business benefits and costs of the technology and may see AR's PoC as a technology study without the necessity for proof, i.e., a production proof version.

With an EU OP (such as KIEMI), the ID types of collaboration are predefined in the OP requirements. The same set of requirements also contains targets for project results which can be related to certain products or services through ID or a target may be related to co-creation activities or to research and development activities between AR and ID. in this project, a production proof version is not included, only PoCs. It is assumed that ID will continue the production proof version from the results of the project.

The model used should take different types of ID into account. It should also take into consideration the fact that innovation activities and technology transfer may happen in all phases or steps. As an example, Punter [8] highlights a case where design work

was able to add value for ID. Similarly, in projects, value can be produced in cases where some commercial product, already designed for a certain usage, has been applied in a new environment through pilot case activities.

Naturally, activities to develop a suitable collaboration model fall mostly to the party responsible for the project, as here on the AR side. The model and its efficiency define success for current and future collaboration between AR and ID.

A study by [10] presents the Certus model, which was developed at a Norwegian research-based innovation center. Their needs for a collaboration model contained similar elements to the DMPP model. They required deeper research knowledge of co-creation activities via problem definition and solving tasks and more active dialog between researchers and practitioners to align their expectations. They also wanted to ensure that the results and outputs from research projects that are created have practical relevance and benefit for their partners and that the results can be transferred and exploited effectively by their partners.

The Certus model [10] contains seven phases, from problem scoping to market research. Whereas the first four phases (problem scoping, knowledge conception, knowledge and technology development, and knowledge and technology transfer) can be regarded as similar to proof-of-concept development, the following three phases (knowledge and technology exploitation, organizational adoption, and market research) are more related to production proof activities.

2.4. The KIEMI project

The reduction of greenhouse gas emissions is one of the most challenging global objectives of the near future. Low carbon emissions, energy savings, a climate-friendly approach, and ecologically sustainable choices require new and innovative services, solutions, and products. One of the biggest potential areas where savings can be made is energy use in properties in Finland. The KIEMI project, carried out by Tampere University Pori unit, designed and developed methods and technologies that aid in finding and achieving the property- and situation-specific ”energy minimum”, i.e., a situation where the minimum amount of energy is used while still preserving a comfortable environment within the building. In the KIEMI project, the primary focus was not on new properties or so-called ”smart buildings”, but on older buildings and apartments that do not contain modern automatic and intelligent devices commonly used for controlling the quality of the living and working environment.

Proof-of-concept demonstrations and prototype applications were developed in the KIEMI project that illustrate how cost-effective, open, and modular solutions can be utilized to improve the energy efficiency of buildings. Further, a decrease in overall energy usage will lead to cost savings related to energy expenses and reduce the carbon footprint caused by, for example, the heating, cooling, and air conditioning of buildings.

In the present world situation in 2023, the theme of the project, energy savings, is especially topical, at least in Europe. The KIEMI project partners consisted of organizations and companies who were able to take part in the pilot cases implemented during the project by providing properties, equipment, sensors, and measurement data or by acting as experts. The results of the project can be utilized by all those involved with the energy and resource efficiency of properties and housing-related wellbeing as well as the relevant private (companies) and public bodies (municipalities).

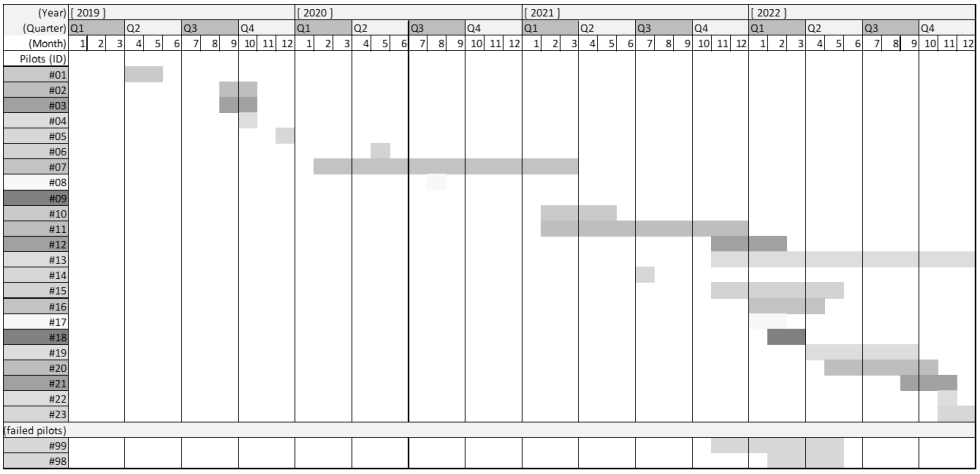

Figure 2. Timeline of pilots in KIEMI.

The commitment of the project partners to the project activities was based on the DMPP collaboration model developed in previous projects. In the KIEMI project, the focal point of the partner-specific co-operation varied, depending on how the partner wished to participate, and how they were able to contribute to the research. Collaboration and contribution to the project pilot cases took place roughly according to the following breakdown:

1. Identifying premises for use in the project (condition measurements in the properties)
2. Handing over existing property data for use in the project (interfaces with existing property measurement systems)
3. Determining measurement needs and planning pilot cases together (tailored needs for condition measurement of the target)
4. General development of condition measurement (developing sensor and measurement systems in collaboration with project partner)

During the project a total of 23 different types of pilot cases were carried out related to the energy efficiency and condition measurement of properties. The pilot cases conducted during the KIEMI project as well as the prototype systems developed for them and the technology testing have been reported extensively in the form of scientific articles (several internationally peer-reviewed research publications). Figure 2 shows the schedule of pilot case implementation by month and quarter over the duration of the project. For interrupted pilot cases, the timetable describes the time interval during which discussion and reflection took place.

3. Process model for prototyping: Descriptive model for the prototyping process (DMPP)

The purpose of this section is to present how the selected process model has supported the work within the projects. Our descriptive software process model for IoT prototyping was introduced in [3]. The DMPP was developed during a previous project where the

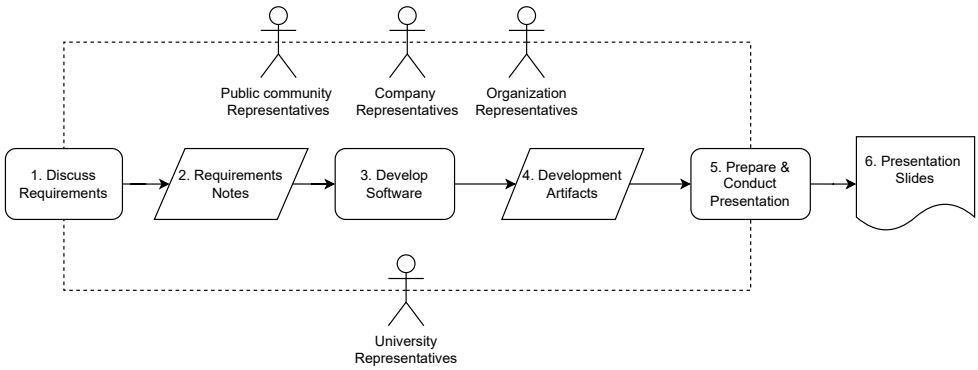

Figure 3. Process model for prototype development. Adapted from [3].

prototyping focused on one area. The DMPP was developed using the descriptive process model (DPM) approach [11]. The basic concepts related to processes are role, activity, resource, and artifact. The example is illustrated by a developer (role) involved in software development (activity) using a programming tool (resource). The activity produces some software (artifact) used in a prototype system. The process data for the model is collected through interviews with the developers involved in the four different prototype development processes. Four prototype development projects and their outcomes were reported in several studies [12], [13], [14], [15]. The common factor in all of the studies is that they present developed IoT prototype systems that gather data.

When the KIEMI project started, we noticed that this DMPP could be an acceptable way to approach the subject. During the project, we actively searched for pilot cases (Step 0) where previously collected knowledge about prototyping IoT data-gathering systems could be used. Figure3 presents the DMPP [3] including steps one to six. The pilot case starts with an issue related to a suitable situation for the research group. The pilot case ends after it has been presented to the customer and other reports have been published. After the pilot case, there is also the possibility to add step 7 (Production proof mentioned in 2.3) which consists of following up the procedure, e.g., client or someone outside of the original pilot case group wishes to utilize the prototype or parts of it. The second possibility is that the developed prototype system goes into production and needs further support (this kind of situation is reported in [14]).

Figure 3 presents the DMPP model. The model includes six steps and the roles, activities, and artifacts can be described as followed using the SW/HW framework [3] and the DMPP [2]:

1. The first step starts from the requirements definition, a collaborative discussion between the developers and the client. The client defines what kind of data would be useful. The developer group starts to define the hardware and overall architecture of the system and how the data will be collected by the software. The selected hardware mostly determines the software environment and tools used. Benefit - Clarification of the problem item together with the customer. Limitation - Does the development team have sufficient expertise in the subject area?
2. The outcome of the discussion is the first artifact: for example, the prototype system requirements in the discussion notes. The developer group constructs the first architecture model of the component interconnections. For example, in IoT

systems, we describe the practice of how to define a system by reusing the system definitions of previous prototypes. Light documentation has been found to speed up stage completion, but may cause problems later if the system is put into production.

3. The third step is the software/hardware prototype development made by the research group including the project manager and SW/HW developers. The IDs' representatives are involved in the development process in the role of instructor. In this step, the SW/HW framework is used as the guideline for selecting the components for the prototype. The SW/HW framework gives guidelines and speeds up development when the operating process of suitable components has at least partially been thought through in advance. Reuse of components also makes it easier when the number of background studies decreases.
4. The fourth step introduces the working prototype artifact, which consists of the developed software and hardware components. Also, the interconnections of the components are tested. The testing process overall is usually only the functional testing of the prototype system. Additionally, the gathered data is inspected and if possible, compared to the expected results. Another notable issue is the fact that, if the system is later put into production, testing must be carried out more thoroughly.
5. The fifth step includes preparing the outcome of the development process. Further, this step includes presenting the prototype and its functionality to the ID. The SW/HW framework can be complemented if necessary.
6. The sixth step is to publish the results, for example, the prototype system, collected data, and analysis of the project. For example, in a university environment, the the publication of results is important for supporting future research projects.

The process model in Figure 3 is a simplified presentation of the prototype development process. It gives abstract instructions for the operation with defined steps to implement the pilot case from start to finish. If all of the steps are performed, the level of the outcome is predictable. The model is sufficient for developing a prototype, and also makes it possible to add more activities if needed. For example, procedures such as iterations, testing, and customer testing could be included in the process. Further, because the model is developed from university pilot cases, it combines two factors: software/hardware prototype development and collaboration with customers. Both of these are discussed in the following section when the usability of the DMPP in the KIEMI project is evaluated.

4. DMPP utilization in the KIEMI project and technology transfer

The purpose of this section is to describe how the DMPP model was utilized in the work process of the KIEMI project. This section also describes how different parties were involved in the project, what kind of collaboration actions were taken during the DMPP steps, and which technology transfer actions occurred during the work process. Figure 4 presents an overall picture of the project, collaboration, and DMPP process in the form of the Business Process Model and Notation (BPMN, [16]).

4.1. Project partners

In the overall picture (in Figure 4) four groups can be recognized in their own swimlane:

1. EU OP and its program documents and goals (via OP documents and goals) must be taken into account for project content and implementation.
2. University within its third mission (TM) and its strategy (via University Strategy) which gives guidelines for research group activities and publishing of project work.
3. Project (like KIEMI) activities are carried by project team members (academic researchers, AR) and activities can be divided into three sub categories:
   (a) Project management (Management) is responsible for implementation of the project plan (Project Plan) and reporting project results to the funding representatives of EU OP (OP supervision) as well keeping track of research publications for university representatives (Research supervision). Project management also acts as the selector of new prototypes in the form of collaboration and pilot case actions.
   (b) DMPP process (DMPP) and its six steps (1-6), which are linked to each other and to collaborative actions with IDs via prototype and pilot case actions.
   (c) Collaboration and Piloting (Collaboration/Piloting) which contains actions and paths supporting DMPP process steps.
4. Collaborative Organization(s) are representatives of collaborating IDs and with whom the content of prototypes and usage via pilot cases is co-created and co-developed.

Technology transfer (and technology creation) takes place between AR and ID via project work and the work process used in it.

4.2. The work process

In Figure 4 the work process of project work can be divided into the following actions (one to eight):

1. The project starts when the project administration (Management) is organized. The project administration defines/selects an appropriate pilot case (Select New Pilot Case), the resources and actions required for the content, and launches the pilot case (Start Pilot Case).
2. From the point of view of the project, a single collaborative pilot case starts (in Collaboration/Piloting) with the invitation of the collaborator (Collaboration Call) and the agreement on cooperation (Collaboration Ignition). For pilot cases #17, #18, #19, and #23, invitations to collaboration IDs were sent via a 3rd party.
3. The first phase of the DMPP process (Discuss Requirements) starts when the project has established contact with the collaborator (ID) and the actual discussion of requirements and objectives begins (Requirement Discussion). For pilot cases #17, #18, #19, and #23, we also received positive responses to collaborate. The project utilizes the discussion base created in previous discussions (Achieved Prototype Pilot Requirement Notes) as a basis for a new discussion. ID brings their views (needs and support and available partners or technical vendors (TV))

to the discussion. For example, needs can be related to certain sensors or measurements and support can be related to the facilities where measurements are made. This starts technology transfer actions between AR and ID/TV. The discussion will result in a decision to continue cooperation and (in a positive decision) the content of the next phase of the DMPP process, namely the requirement notes (Prototype Pilot Requirement Notes).

As the discussion produces a positive decision (OK To Initiate Prototype Pilot?), a pilot case (Prototype Pilot Ignition) and the third phase of the DMPP process (Develop Software) will begin (Start Prototype develop). On the ID side, the corresponding decision (OK To Initiate Prototype Pilot?) to proceed initiates support for prototype development and supports prototype piloting activities. In the event of a discussion producing a negative decision (or cooperation ending without successful agreement), the pilot case is reported to the administration as interrupted (Pilot Case Aborted), which then processes the interruption result. For pilot cases #98 and #99, collaboration was ended in the first phase of the DMPP process (Discussion).

§

4. In the third phase of the DMPP process (Develop Software), the prototype artifacts (software and hardware) needed in the pilot case are developed. The development of the prototype (Develop Prototype (SW/HW)) is guided by the requirements recorded in the previous phase (Prototype Pilot Requirement Notes in Requirement Notes) and utilizes any artifacts (Development Artifacts) that may have been generated in previous cases. Prototype development involves discussions and exchanges of information (Technical Discussion) with the ID and TV brought into the pilot case. New and advanced artifacts resulting from the prototype development phase are introduced to artifact management (Manage Artifacts in Development Artifacts), representing the fourth stage of the DMPP process. Pilot case #11 was an example of a case where both technology creation and technology transfer occurred between AR and ID.
5. The completion of the prototype development phase (Prototype Develop Ready) initiates the prototype pilot case execution phase (Execute Prototype Pilot in Collaboration/Piloting), where pilot case data and results are collected from the use of the prototype at the pilot case site (received from ID). The data collected in the prototype pilot case is included/added to the Development Artifacts (via Manage Artifacts) generated in the third step (Development Software).

   The piloting of a single prototype could take several weeks. For pilot case #19, data was collected for a period of several months and data collection was monitored online. On the other hand, pilot case #13 contained data for a period of over one year and data was collected afterwards from ID's database. The latter case also contained technology transfer between AR and ID to tune up ID's interface about database metadata information.
6. At the end of the prototype pilot case (Start Prototype Presentation), the penultimate stage of the DMPP process, the preparation phase for the presentation of the results is initiated. In this phase (Prepare Presentation in Prepare & Conduct Presentation), the artifacts generated during the prototype pilot case are compiled (via Manage Artifacts in Development Artifacts) into presentation materials for the final stage of the DMPP process (via Manage publications in Presentation

Slides) and the presentation of the materials to ID (Conduct Presentation in Prepare & Conduct Presentation). In the preparatory phase, previous presentation materials (Archieved Slides via Manage Slides) can be utilized. The presentation schedule is discussed with ID (Call For Presentation) who gathers their team and TV for the meeting (Receive Presentation in Collaborative Organizations(s)). The presentation ends steps five and six of the DMPP process for collaboration tasks (Prototype Presentation Ready). Pilot cases #17, #18, #19, and #23 were examples of technology transfer via a presentation and delivered report documents. Case #23 also included a representative from ID's TV side.

7. There is usually a feedback discussion (Ask Feedback/Give Feedback in Collaboration/Piloting) following the presentation (Prototype Presentation Ready) on the results obtained from the use of the prototype and the implementation of its piloting, as well as on the success of the collaboration. Feedback processing concludes the collaborative pilot case (Pilot Case Ready) and technology transfer actions between AR and ID/TV. Pilot case #10 contained a feedback discussion where ID felt that the collaboration was very successful and they requested another pilot case (#16 in the list) after the issue for the target facility had been solved thanks to the first pilot case.
8. At the end of the pilot case (Pilot Case Ready), the information is sent to the administration (Pilot Reporting), which records the project indicators and progress (via Project Indicators) for reporting to the EU OP financier (OP Supervision) on the pilot case. The administration is also responsible for sharing the research results (Research Reporting) through communication channels (via Project Publications) and to the university (Research supervision via Research Publications). Actions for communication tasks are also reported to the EU OP financier (OP Supervision).

   Artifacts and publication slides generated in the DPMM process may be published or distributed in connection with the news blog. Pilot cases #17, #18, and #19 were examples of (one way) technology transfer via news blogs for any other ID or individual interested in the topic.

   When a single collaborative pilot case has ended, management decides on the need for another pilot case (Is Project Completed?). Once the required number of prototypes and their piloting work have been completed (or project time is coming to an end, it leads to the final tasks and the end of the project.

5. Pilot cases in KIEMI

The purpose of this section is to present the background or characteristics related to the pilot cases (comparison table) as well as to compare the activity levels of collaboration associated with the pilot cases.

Table 1 contains pilot case specific reference parameters. **Pilot cases** are numbered with a running identification number according to their starting time (see pilot case timeline in Figure 2). Comparative data has been compiled for each pilot case using six parameters. The **User Group** parameter describes the classification of the piloting target. Options include company (A), public operator (B), entity (C), and others (D). The **Stakeholders** parameter describes the classification of parties who joined the piloting target.

Table 1. Properties of pilot cases in the KIEMI project

Pilot case	User group (A/B/C/D)	Stakeholders (E,F,G,H)	DMPP usage (1-6 / 6)	OTS used (Yes/No)	Published content (-,X,Y)	Collaboration activity level (Low/Mid/High)
#01	B	E	3	No	-	LOW
#02	B	E	3	No	-	LOW
#03	B	E	5	Yes	X	LOW
#04	B	E	3	Yes	-	LOW
#05	B	E	5	Yes	-	LOW
#06	D	E	3	Yes	-	LOW
#07	B	E+G	3	No	Y	LOW
#08	D	E	5	Yes	-	MID
#09	D	-	6	No	X	LOW
#10	B	E+F+G	6	Yes	X+Y	HIGH
#11	B	E+G	6	No	X+Y	HIGH
#12	A	E+F+G	5	No	Y	MID
#13	A	E+G	5	No	Y	MID
#14	D	E	5	Yes	-	MID
#15	B	E+F+G	5	Yes	X	LOW
#16	B	E+F+G	6	Yes	X	LOW
#17	C	E	5	Yes	Y	LOW
#18	C	E	5	Yes	Y	LOW
#19	C	E	5	Yes	Y	MID
#20	A	E	6	Yes	-	MID
#21	A	E	3 (*)	Yes	- (*)	? (*)
#22	D	E+F	4	Yes	-	LOW
#23	A	E+F	5	Yes	Y (*)	MID
(failed pilots)						
#99	A	E+G	2	N/A	-	-
#98	A	E+G	2	N/A	-	-

Alternatives include subscriber (E), users (F), technical vendor (G), and developer (H). Several parties may have been involved in the piloting. The **DMPP usage** parameter describes the number of steps in the DMPP process utilized at the piloting site. Each pilot case may have utilized one or more, or even all of the steps. The **OTS used** parameter contains information on whether off-the-shelf components were used in the pilot case. The **Publish content** parameter includes information on whether the results of the pilot case were released in a transparently available format through a research publication (X) or project news blog (Y) or both. Some pilot case results were only handled internally. The **Collaboration activity level** parameter describes the collaboration activity of ID during the work process (in Fig. 4). For a couple of pilots some information was not yet available during the writing of this paper and that information is marked with (*).

5.1. Pilot cases with high-level collaboration

In high-level collaboration, the counterpart (ID) demonstrates active cooperation at all stages of the work process. ID brings to the discussion stage a view of the features required for the prototype and its operating environment. ID also demonstrates its interest in the technical content of the prototype resulting from the development phase and is involved in the processing of observations made during the pilot case phase. In high-level cooperation, ID shows interest in the content of the results (report) and highlights their views on the exploitation of the results. It is clear that ID benefits from high-level collaboration in many ways.

Pilot case #10 is a good example of high-level collaboration. The target was a day-care center, which had received feedback about poor air quality inside the building. The first target was to measure the temperature, humidity, and CO2 values at different times and report the readings to the partner. The first results showed that at certain moments the temperature and CO2 values had risen. During the early phase meeting where the results were shown, we decided with the partner(ID) to continue and expand the pilot case. Expansion meant contacting the air conditioning equipment supplier(TV). This gave us an interface with the air conditioning system. In addition, they expanded the sensor number and type to collect data that was more specifically environmental. Our project team also used the previously developed visualization tool to this pilot case.

Outcome: This was the widest pilot case with several partners(TV and ID), using previously used and developed components.

5.2. Pilot cases with mid-level collaboration

In mid-level collaboration, the counterpart (ID) is involved at the beginning and end of the work process and in some way also involved in the development content of the work process. ID support may be required, particularly in situations where part of the prototype content is sourced from an ID-managed data source. In general, ID benefits from mid-level collaboration, at least from the perspective of external testing obtained for its own functions.

Pilot case #13 can be used as an example of mid-level collaboration. In this case ID had a vast amount of facilities at their disposal and they had already implemented a data sensor system and were using data analysis tools via their TV. For the pilot case, ID allowed AR to use their data (collected by ID's TV) for AR's tools to produce another kind of analysis from the data. ID did not participate in the actual SW development, but the use of data via ID's API during piloting required technical discussions. The benefit for ID from the piloting case was related to experience gained about their API and the knowledge received via the pilot case report.

5.3. Pilot cases with low-level collaboration

In low-level collaboration, the counterpart (ID) is involved in the work at the beginning (Discuss Requirements) and end of the process (Presentation Slides). In these cases, the project team has most often conducted a search for actors interested in collaboration and provided the test target, giving the ID the opportunity to obtain new information about its application through the report. Thus, AR also provides technology transfer to ID. For a project, low-level collaboration can also be beneficial. Piloting over a longer time period does not necessarily burden the project staff and the results obtained from the pilot case can be very useful for demonstrating the functionality of the prototype.

Low-level collaboration is also no obstacle to publicizing the results of the project - on the contrary, for example pilot cases #17, #18, and #19 (entities as user groups) and the disclosures generated from their results have contributed to the local visibility and reputation of the project. The presentation materials have also been utilized to obtain new, higher-level collaborative cases.

5.4. Failed pilot cases

In addition to the above levels of collaboration, it is also useful to point out exceptions where piloting collaboration ended or was interrupted. In the work process, piloting can usually be interrupted only in its initial stages.

The reason may be ID's reluctance (or resource shortage) to initiate collaboration. ID is not interested even in free piloting if it does not promise immediate benefit; in practice, however, that requires some involvement. Piloting may involve TV on ID's part, which is necessary but TV is reluctant (similar to ID's own reluctance).

Another reason may be that something comes up during the discussion stage (Discuss Requirements) that makes it impossible to continue or not meaningful to continue the piloting.

Even after progressing to the technical stage of the DMPP process (Develop Software), a situation may arise where a developed prototype is found to be unworkable. From the point of view of collaboration, the work process is interrupted, although from the point of view of research, a non-working prototype is also part of the results of the research. If the idea works, the hardware can be replaced with more suitable hardware in the next iteration round.

Pilot cases #98 and #99 are examples of cases where collaboration was interrupted. In case #99, ID was interested in collaboration, but access to required data was managed via ID's TV's API and TV had little or no interest in collaboration. For case #98, ID was also interested in collaboration. During the discussion stage AR noticed that it would be too difficult to produce data in such a form that would work for ID's needs. In both cases proceedings (in discussion stage) were paused and finally project management decided to shelve the piloting case.

It is worth mentioning that in the work process there were also some cases where project management was asked to help to communicate with ID to make sure that the collaboration would continue. Interruptions in collaboration cause serious harm to the work process. For example, due to material limitations, when the test equipment is reserved at one site, the next piloting target cannot be handled.

6. Usability and evaluation of DMPP in the KIEMI project

The DMPP was developed for the production of prototypes at the university. The goal has always been to produce scientific results from the prototypes. The research group is from non-commercial institutions and therefore the focus is not on achieving financial goals. This subsection clarifies the advantages of different phases of the DMPP. The KIEMI project used the DMPP model to create prototypes together with collaborative partners. This project and its approach to the subject through prototyping demonstrated the functionality of the DMPP model, especially in prototyping projects like this one. The suitability of the different phases of the DMPP model can be assessed through the KIEMI project pilot cases as follows:

Discuss requirements: Most pilot case projects involve an external partner(ID) when discussing objectives. The level of collaboration varies a lot. In low-level collaboration e.g., in pilot cases #19 and #22, the partner provided the premises to perform the measurements. The partner does not make any special requests. The output for the part-

ner is a report which may lead to further actions. If the collaboration is closer, as when the partner takes part in further discussions, the starting point is also directed more by the partner. In these cases, the partner mostly has some issue which should be researched, e.g., they have been notified of poor indoor air quality (pilot case #10). Usually in these cases, the original task assignment expands during the pilot case and more partners join in. The DMPP is suitable for this kind of activity because the non-commercial leader – the university research team – is focused on research goals rather than financial goals. Further, the additional research/technical goals set by partners are shown to be applicable to the operation of the model within the iteration rounds. The best example of this kind of activity is pilot case #10 where the university research team led the pilot case and collected the necessary partners (e.g., ventilation technology supplier and building caretaker).

Requirements notes are an important part of documentation and their main purpose is to guide the pilot case in the selected direction. The usage of the DMPP shows the advantage of ”light documentation” for getting things started; the usage of previously defined architecture models and device configurations also speeds up the operation. The term ”light documentation” also means the reuse of the technological choices and definitions made in earlier pilot cases. The exception is pilot case #23, where the final report included a section on desired goals. Internal requirements are also mentioned in several cases, e.g., the research group wants to change or update some specific feature. The ”light documentation” idea is based on the ”Some Things Are Better Done than Described” [18]. Light documentation and process modeling is focused on the university and other research institution environments where the aim was prototyping rather than the development of commercial products. Of course, this leads to a larger amount of work if technology transfer to some partner starts from the prototype.

The **Develop software** phase uses the artifacts of previous requirements as a loose guideline. For example, UI [19] and backend [20] software developed in pilot case #09 were used in all subsequent pilot cases (excluding #11). In the DMPP, changes to the requirements are possible if it is seen to be of some benefit. Further, the requirement changes were not normally discussed with partners unless something was needed from them. The DMPP does not set requirements for the software or hardware components used, but we noticed that the usage of off-the-shelf components accelerated prototype development. The second advantage of these kinds of components is the ability to vary the prototype solutions when we have to conform to the requirements of the selected components.

Development artifacts are typically fully working prototype systems which are also the main goals of this phase for the DMPP. In the KIEMI project, this phase usually involved installing the prototype to collect data at a target provided by the partner. Most of the prototypes were working SW/HW prototypes, but there were also only SW prototypes for analyzing and visualizing the customer’s collected data (#12 and #13). The main purpose of the DMPP is to produce a working prototype and therefore only the main functions of the prototype are utilized. Additionally, the documentation or testing could be done only partially. This kind of approach speeds up the development but could slow down the technological transfer later on.

The **Prepare & conduct presentation** phase is for reporting the results. In longer projects we noticed that the document reuse of skeleton reports accelerated this phase. In pilot cases #20 and #23 of the final phase of the KIEMI project we collected a skeleton

report from pilot case #19. This automation sped up the reporting phase. This shows that when using the DMPP model, reporting will mostly include the same components.

Presentation and publishing of the results are the last phase in the DMPP. In successful pilot cases the partners are usually interested in further developing the prototype and the technology transfer will continue from this point. One significant advantage of the DMPP is the ultimate purpose of publishing the scientific material (pilot cases #03, #09, #10, #11, #15, and #16 have been published) and other public material from the pilot cases.

Overall analysis and DMPP's suitability for projects were shown in the KIEMI project. Two approaches were used in the project: the software development style and collaboration style. The DMPP is able to connect both styles. The project was shown to be successful for university-enterprise (AR-ID) collaboration in the context of prototype development. Further, based on the results in creating usable prototypes, the model can be seen as a success.

7. Conclusions

RQ1: Collaboration. **How was university-enterprise collaboration executed in practice using the DMPP?** The DMPP process was part of a project (Fig. 4) where the content was guided by the objectives set for the project (Management) and an individual prototype was made through collaboration (Collaboration/Piloting). The DMPP process was in the background (invisible to ID), but it was able to provide support for collaboration (AR-ID) through all of its six phases. The ability of the DMPP process to support technology transfer was highlighted in phases 1, 3, 4, and 5.

For Step 2 (Requirement Notes), the content was usually only left up to the project team (AR). Regarding companies (ID and their TV), it is unknown whether they had one of their own similar methods in place. At the very least, communication (emails) enabled ID (and their TV) to receive and store requirement-related data.

As far as Step 6 is concerned, ID received a report on the content and results of most pilot cases. For pilot cases where content was distributed through open channels (such as Project news blogs and Github in Presentation slides), ID (and TV) had the opportunity to catch up, not only with their own content, but also the content of other pilot cases.

The collaboration also demonstrated that university and corporate representatives have a very different view of technology, and therefore of pilot cases as a whole. Especially in small companies, the desire and ability to recognize the value and benefits contained in the prototype is often low, and the university needs to convince the collaborator of the benefits of a prototype that requires effort on their part.

In a longer-term project, it should be considered whether each prototype is intended for actual technology transfer or whether that stage will only come when satisfactory prototypes have been achieved. In practice, the project requires that pilot cases at the beginning of the project are conducted mainly with organizations offering test environments and only at the end does the content begin to involve technology transfer.

There was no investment in cost calculations or business models in the design of university prototypes and this may have contributed to the amount of interest shown by companies. To improve collaboration it is good to add a point where the company provides a (suitable general level) assessment of the prototype as well as the associated re-

turn on investment (ROI). With the feedback received, the research team would accumulate expertise in designing the next prototype and opportunities to produce a result that is of more interest to the company. The ability to produce prototypes valued by companies is a significant strength and advantage for a university operator that organizes projects. It is also an advantage for future project partner searches.

RQ2: Reusability. **How did the reusability of the artifacts in the DMPP steps support the workflow of the pilot case?** The use of the DMPP model led to the reuse of artifacts when the mode of operation remained the same even though the pilot cases changed. In the prototypes, we mainly used the same software and hardware components that had been used before. Further, we also always tried to introduce some new components , because this increased knowledge and expanded component-based variation. The DMPP uses light documentation to speed up prototype development, but we noticed that separate phases in different pilot cases started to contain the same type of documents. Therefore, the conclusion is that the DMPP leads to re-use of skeleton documents in different pilot cases.

The findings of the research presented above represent the context of a Finnish university and it would require more research to obtain universally applicable results. However, these observations and findings provide the basis for the possibility to extend the research to an external comparison between universities in different countries.

8. Summary

This article focused on the KIEMI research project conducted at the Pori Unit of Tampere University during 2019-2022. The project used the earlier developed Descriptive Model of Prototyping Process (DMPP) to guide university-enterprise collaboration. The project consisted of several pilot cases and prototypes, which were made in collaboration with companies, and offered real-world problems. This article reviewed and evaluated the suitability of the DMPP for this topic. The article dealt with the collaboration between university and enterprises, and reusability within the DMPP. The paper presented several pilot cases made in KIEMI, and described the usage of the DMPP. Finally, the paper evaluated the model, presented some of the challenges faced, and discussed future research topics.

Acknowledgements

This work is part of the KIEMI project and was funded by the European Regional Development Fund and the Regional Council of Satakunta.

References

[1] Saari M, Sillberg P, Grönman J, Kuusisto M, Rantanen P, Jaakkola H, et al. Reducing Energy Consumption with IoT Prototyping. Acta Polytechnica Hungarica. 2019;16(9, SI):73-91.

[2] Saari M, Rantanen P, Hyrynsalmi S, Hästbacka D. In: Sgurev V, Jotsov V, Kacprzyk J, editors. Framework and Development Process for IoT Data Gathering. Springer International Publishing; 2022. p. 41-60. Available from: https://doi.org/10.1007/978-3-030-78124-8_3.

[3] Saari M, Soini J, Grönman J, Rantanen P, Mäkinen T, Sillberg P. Modeling the software prototyping process in a research context. In: Tropmann-Frick M, Thalheim B, Jaakkola H, Kiyoki Y, Yoshida N, editors. Information Modelling and Knowledge Bases XXXII. vol. 333. IOS Press; 2020. p. 107-18.

[4] Vorley T, Nelles J. Building Entrepreneurial Architectures: A Conceptual Interpretation of the Third Mission. Policy Futures in Education. 2009 6;7:284-96. Available from: http://journals.sagepub.com/doi/10.2304/pfie.2009.7.3.284.

[5] Zomer A, Benneworth P. The Rise of the University's Third Mission. Reform of Higher Education in Europe. 2011:81-101. Available from: http://link.springer.com/10.1007/978-94-6091-555-0_6.

[6] Basili V, Briand L, Bianculli D, Nejati S, Pastore F, Sabetzadeh M. Software Engineering Research and Industry: A Symbiotic Relationship to Foster Impact. IEEE Software. 2018 9;35:44-9. Available from: https://ieeexplore.ieee.org/document/8409904/.

[7] Salomaa M, Charles D. The university third mission and the European Structural Funds in peripheral regions: Insights from Finland. Science and Public Policy. 2021 jul;48(3):352-63. Available from: https://academic.oup.com/spp/article/48/3/352/6126876.

[8] Punter T, Krikhaar RL, Bril RJ. Software engineering technology innovation–Turning research results into industrial success. Journal of Systems and Software. 2009;82(6):993-1003.

[9] Arza V, Carattoli M. Personal ties in university-industry linkages: a case-study from Argentina. The Journal of Technology Transfer. 2017 8;42:814-40. Available from: http://link.springer.com/10.1007/s10961-016-9544-x.

[10] Dusica M, Arnaud G. Industry-Academia research collaboration in software engineering: The Certus model. Information and Software Technology. 2021 4;132:106473. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0950584920302184.

[11] Becker-Kornstaedt U, Webby R. A comprehensive schema Integrating Software Proces Modeling and Software Measurement. IESE-Report No 04799/E. 1999.

[12] Grönman J, Rantanen P, Saari M, Sillberg P, Vihervaara J. Low-cost ultrasound measurement system for accurate detection of container utilization rate. In: 2018 41th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE; 2018. .

[13] Soini J, Sillberg P, Rantanen P. Prototype System for Improving Manually Collected Data Quality. In: Budimac Z, Galinac Grbac T, editors. Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2014, September 19-22, 2014, Lovran, Croatia. Ceur workshop proceedings. M. Jeusfeld c/o Redaktion Sun SITE; 2014. p. 99-106.

[14] Soini J, Kuusisto M, Rantanen P, Saari M, Sillberg P. A Study on an Evolution of a Data Collection System for Knowledge Representation. In: Dahanayake A, Huiskonen J, Kiyoki Y, editors. Information Modelling and Knowledge Bases XXXI. vol. 321. IOS Press; 2019. p. 161-74.

[15] Grönman J, Sillberg P, Rantanen P, Saari M. People Counting in a Public Event—Use Case: Free-to-Ride Bus. In: 2019 42th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE; 2019. .

[16] Object Management Group, Inc. Business Process Model and Notation; 2023. Accessed January 13, 2023. Available from: https://www.omg.org/spec/BPMN/2.0.2/About-BPMN.

[17] Janne Harjamäki. Technology transfer in the Kiemi project; 2023. Accessed January 23, 2023. Available from: https://cawemo.com/share/bb6b8086-13b7-4ab9-bb86-92cdaf9a5d18.

[18] Hunt A, Thomas D. The Pragmatic Programmer. Addison-Wesley; 2000.

[19] Nurminen M, Lindstedt A, Saari M, Rantanen P. The Requirements and Challenges of Visualizing Building Data. In: 2021 44th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE; 2021. .

[20] Nurminen M, Saari M, Rantanen P. DataSites: a simple solution for providing building data to client devices. In: 2021 44th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE; 2021. .

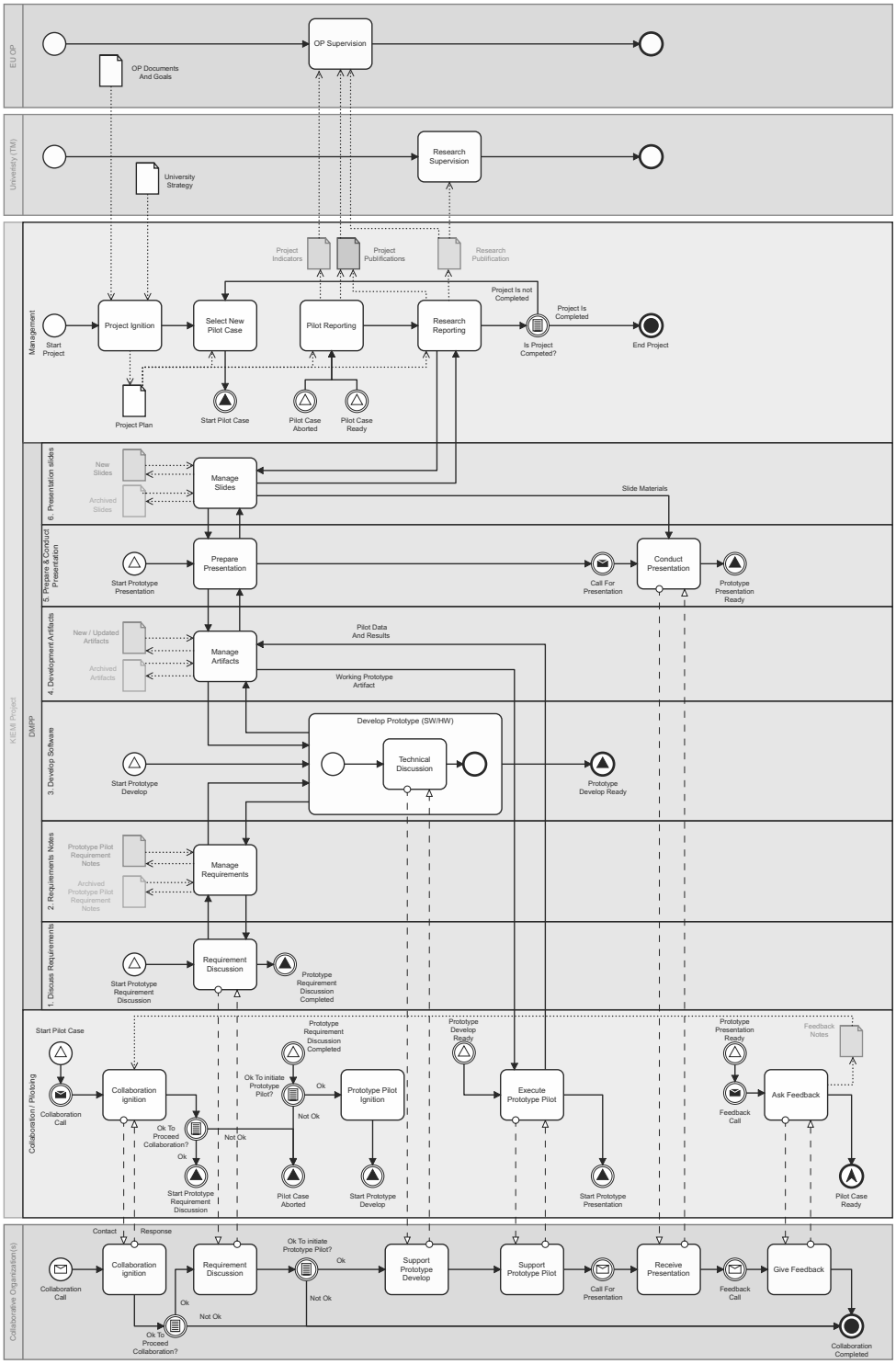

Figure 4. Technology transfer in the Kiemi project. (The figure is available in [17])