PEKKA SILLBERG

Development of New Data Processing Model for Data and Software Reusability

TAMPERE UNIVERSITY 2024

Tampere University

PEKKA SILLBERG

Development of New Data Processing Model for Data and Software Reusability

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Management and Business of Tampere University,
for public discussion in the auditorium 125
of the University Consortium of Pori, Pohjoisranta 11 A, Pori,
on 12 January 2024, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Management and Business
Finland

Responsible supervisor and Custos	Professor Kari Systä Tampere University Finland	
Supervisors	DSc (Tech) Jari Soini Tampere University Finland	Professor Emeritus Hannu Jaakkola Tampere University Finland
Pre-examiners	Professor Boštjan Brumen University of Maribor Slovenia	Professor Ajantha Dahanayake LUT University Finland
Opponent	Professor Markku Tukiainen University of Eastern Finland Finland	

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

ISBN 978-952-03-3215-0 (print)
ISBN 978-952-03-3216-7 (pdf)
https://urn.fi/URN:ISBN:978-952-03-3216-7

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

PunaMusta Oy – Yliopistopaino
Joensuu 2023

PREFACE

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 by Tampere University of Technology, and by its successor, Tampere University, engaging in various and interesting research projects. Financial support for this thesis was provided by the Satakunta Regional Fund of the Finnish Cultural Foundation, and the High Technology Foundation of Satakunta.

I had the privilege of being guided by professor Emeritus Hannu Jaakkola, DSc (Tech) Jari Soini, and professor Kari Systä. Hannu, your assistance, contribution, and mentorship go beyond any measurable amount. I cannot imagine where I would be without your influence on my journey so far. Jari, thank you for your endless patience and support, as well as your ability to find time and resources for my thesis work. Kari, you were always able to identify even the tiniest inconsistencies and found ways to improve the clarity of the thesis. Thank you all for believing in me. Additionally, I am very grateful to professors Boštjan Brumen and Ajantha Dahanayake for their excellent service as pre-examiners of this thesis.

I want to express my gratitude to my colleagues at Tampere University, co-authors, friends, and everyone else who got involved in my lengthy project. I also want to acknowledge professor Sami Hyrynsalmi, who briefly served as my responsible supervisor and provided the tip to use this particular LaTeX template[1].

The importance of family cannot be emphasized too much. Thank you all for remembering me. My parents remembered to ask me now and then how the thesis work was going. My wife, Chalisa, who has already earned her PhD, was the most important person in supporting and cheering me up. Her parents were very kind in trusting that I would finish the thesis someday. Special recognition goes to little Pekko. Luckily for you, there happens to be at least one picture with a car in this book.

Many things have changed, but one concrete landmark has remained relatively stable over these years. That thing is the former cotton factory *Porin Puuvilla*, located on the north bank of the Kokemäenjoki river in the City of Pori. In fact, it turned out that Puuvilla has become quite an integral and meaningful part of my life. Perhaps even a one-stop service center: receiving a master's degree, a workplace, a place for health and dental care, a thriving shopping center, finding the love

1 classicthesis, https://ctan.org/pkg/classicthesis

of my life and subsequently getting married, and now the upcoming doctoral degree—what else Puuvilla could offer me in the future?

Pori, Finland, December 2023
Pekka Sillberg

ABSTRACT

This thesis studies the challenges in improving the quality of data processing in information intensive software applications, particularly in the practices of software engineering involved in designing and implementing a system model for the various needs of information processing. The research builds upon the traditional methods found in the software engineering literature to create a new data design pattern style for a new data environment. At the same time, as the amount of data keeps growing, the logical management of data (and the various data sources) is becoming more important. Prior research has shown that there is a need for maintainable and systematic ways to manage the produced data more efficiently. How can and should data be managed by the software? To address the aforementioned question, a comprehensive solution will need to apply the disciplines of data management and software engineering together.

The research in this thesis integrates the aforementioned features together in one package, which includes features from software frameworks, design patterns, and architectural styles. The research adopts Design Science Research Methodology in carrying out the research activities, and iteratively refines and evaluates the intermediate results.

The main contribution is the introduction of a conceptual and generic data processing model, which is built on the metaphor of a streaming water apparatus consisting of faucets, sinks, and drains. The main point of the model is how the model treats all data sources equally, and as simply and generically as possible. The generic data source management of the model will be the key on improving the reuse of source code as well as reuse of data. The secondary contribution is a solution derived from the model which provides a reference architecture, definitions, and specifications to realize a generic and reusable software framework.

The work is validated through several architectural iterations and prototype implementations. The framework was implemented and tested in an experimental prototype system with a few use cases. Finally, the findings, and theoretical and practical prospects of the model are discussed. The demonstrated proof-of-concept experiments indicate that the proposed model is feasible solution for systematically manageable data processing, and can improve the quality and reuse of both software and data.

CONTENTS

I DOCTORAL DISSERTATION

1 INTRODUCTION 3
- 1.1 Research Area 6
- 1.2 Research Goals 7
- 1.3 Research Methodology 8
- 1.4 Publications 11
- 1.5 Thesis Structure 14

2 BACKGROUND 15
- 2.1 Implementation of Literature Review 15
  - 2.1.1 Initial Review 15
  - 2.1.2 Generic Review 16
- 2.2 Related Studies 19
  - 2.2.1 Data Management 19
  - 2.2.2 Similar Approaches 21
- 2.3 Software Standards and Best Practices 25
  - 2.3.1 Standards 25
  - 2.3.2 Best Practices 27
- 2.4 Conclusion of the Background Study 29

3 TOWARD GENERIC DATA MANAGEMENT 31
- 3.1 In Search of Data Management 31
  - 3.1.1 Quality Aspects 32
  - 3.1.2 System Models 34
- 3.2 Publications in Detail 36

4 THE DATA PROCESSING MODEL 41
- 4.1 Faucet-Sink-Drain Model 41
  - 4.1.1 Overview 41
  - 4.1.2 Abstraction 43
  - 4.1.3 Implementation 47
- 4.2 Use Cases 50
- 4.3 Prototype System 54
  - 4.3.1 Overview 54
  - 4.3.2 Detailed Description 56
  - 4.3.3 Reusing Technical Constructs 60
- 4.4 Conclusion of the Empirical Study 63

5 DISCUSSION 65
- 5.1 Revisiting the Research Questions 65
- 5.2 Contributions of the Thesis 66
- 5.3 Evaluation of the Constructs 68
- 5.4 Threats to Validity 72
- 5.5 Future Work 73

6 SUMMARY 75

REFERENCES 77

II ORIGINAL PUBLICATIONS

P1 PUBLICATION I 87

P2 PUBLICATION II 95

P3 PUBLICATION III 105

P4 PUBLICATION IV 113

P5 PUBLICATION V 125

P6 PUBLICATION VI 139

LIST OF FIGURES

Figure 1.1 Total data volume worldwide 2010–2024 3
Figure 1.2 Applied DSRM process model 10
Figure 2.1 Overview of Datoms 22
Figure 2.2 ISO/IEC 25012 Target domain 26
Figure 3.1 Product quality model 32
Figure 3.2 Deployment of prototype system 35
Figure 4.1 Conceptual model of data processing 42
Figure 4.2 Data acquisition 44
Figure 4.3 Data storage 45
Figure 4.4 Data identification and filtering 45
Figure 4.5 Data processing 46
Figure 4.6 Data visualization 46
Figure 4.7 Reference architecture 47
Figure 4.8 Model of data flows 53
Figure 4.9 Overview of prototype data flows 54
Figure 4.10 Structure of an example system 62

LIST OF TABLES

Table 2.1 Initial literature review 16
Table 2.2 Generic literature review 18
Table 3.1 Summary of research contribution 36
Table 4.1 Summary of terminological changes 49
Table 4.2 Components of an example system 61
Table 5.1 Analysis of the source code 67

LISTINGS

Listing 2.1 MongoDB Aggregation pipeline 23
Listing 4.1 Initialization of the system 57
Listing 4.2 Example use of Stream Operators 59
Listing 4.3 Example of result output 60

ACRONYMS

API	Application Programming Interface
CSV	Comma-separated Values
DBMS	Database Management System
DIKW	Data-Information-Knowledge-Wisdom
DMM	Data Management Maturity
DMS	Data Management System
DS	Design Science
DSRM	Design Science Research Methodology
DW	Data Warehouse
FAIR	Findable, Accessible, Interoperable, Reusable
FRIEDA	Flexible Robust Intelligent Elastic DAta
GDMSA	Generic Data Management System Architecture
HTML5	HyperText Markup Language 5
IoT	Internet of Things
IS	Information Systems
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MCERTS	Monitoring Certification Scheme
MDMS	Microblogs Data Management System
MDS	Manageable Data Sources
MIME	Multipurpose Internet Mail Extensions
MVC	Model-View-Controller
RDM	Research Data Management
rHMEI	River Heavy Metal Evaluation Index
RO	Research Objective
RQ	Research Question
SQL	Structured Query Language
SE	Software Engineering
SQuaRE	Software product Quality Requirements and Evaluation
UI	User Interface
USDMS	Universal Simulation Data Management System

LIST OF PUBLICATIONS

[Publication I] J. Soini, P. Sillberg, and J. Raitaniemi, "Utilizing adaptive software to enhance information management," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 6, no. 12, pp. 1553–1558, Dec. 2012.

[Publication II] 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)*, Lovran, Croatia, Sep. 2014, pp. 99–106.

[Publication III] J. Soini, P. Sillberg, P. Rantanen, and J. Nummela, "Portable sensor system for reliable condition measurement," in *Proceedings of the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, May 2016, pp. 1397–1402. DOI: 10.1109/MIPRO.2016.7522320.

[Publication IV] P. Sillberg, C. Veesommai, J. Soini, and H. Jaakkola, "Web-User-Interface System Utilizing rHMEI and Open Data for a Water Quality Analyzer," in *Information Modelling and Knowledge Bases XXIX*, ser. Frontiers in Artificial Intelligence and Applications. Netherlands: IOS Press, Feb. 2018, pp. 420–428, ISBN: 978-1-61499-833-4. DOI: 10.3233/978-1-61499-834-1-420.

[Publication V] P. Sillberg, "Toward Manageable Data Sources," in *Information Modelling and Knowledge Bases XXX*, ser. Frontiers in Artificial Intelligence and Applications. Netherlands: IOS Press, Feb. 2019, pp. 101–111, ISBN: 978-1-61499-933-1. DOI: 10.3233/978-1-61499-933-1-101.

[Publication VI] P. Sillberg, M. Saari, J. Grönman, P. Rantanen, and M. Kuusisto, "Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data," in *Intelligent Systems: Theory, Research and Innovation in Applications*, R. Jardim-Goncalves, V. Sgurev, V. Jotsov, and J. Kacprzyk, Eds., ser. Studies in Computational Intelligence. Springer, Cham, Mar. 2020, vol. 864, pp. 99–119, ISBN: 978-3-030-38703-7. DOI: 10.1007/978-3-030-38704-4_5.

Part I

DOCTORAL DISSERTATION

1

INTRODUCTION

The amount of global information has been increasing at an exponential rate, and the trend appears to be continuing. Forecasts predict (e.g., Statista, 2020; Patrizio, 2018; Marr, 2018) that the amount of data is doubling approximately every two years, and by 2024 the amount of data will reach 149 zettabytes. In terms of data volume, the billions of Internet users are now generating the majority of the content (e.g., photos, videos, emails, documents), but in terms of data velocity, the various technical actors such as Internet services and Internet of Things (IoT) devices are leading. The increase of unstructured information among other factors contribute to the expansion of the variability of data. These three characteristics: volume, velocity, and variety, have been identified as the defining features of big data (Laney, 2001).

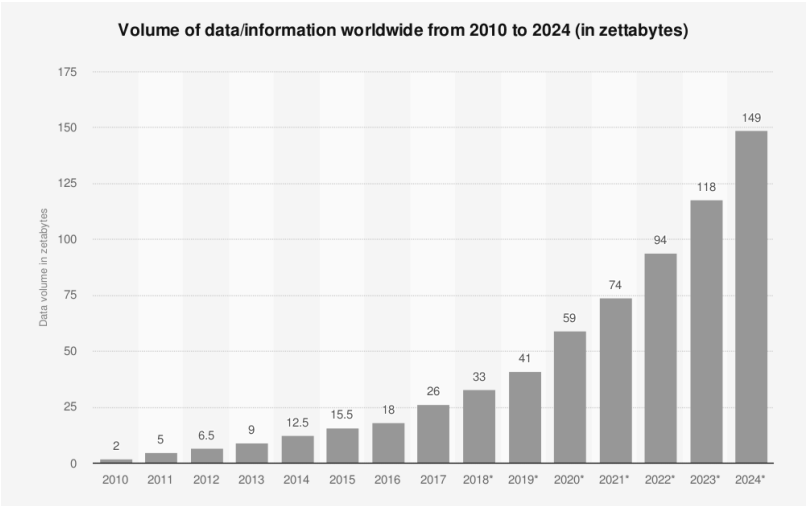


Figure 1.1: Total data volume worldwide 2010–2024. (Statista, 2020)

The challenges of big data extend beyond the physical characteristics (e.g., storage requirements or raw computing capacity). If the amount of data grows as predicted (Figure 1.1), it might not be possible to keep up with the demand by increasing and purchasing more computational resources. Therefore, the logical management of big data by software (e.g., applications and systems) will become more important. At the same time, the complexity of Information Systems (IS) are growing, and consequently system models will grow too, and

focus must be placed on a system of systems (Jaakkola and Thalheim, 2021). It will become more challenging to manage the data modeling of complex, pipeline-based, and/or mixed data sources and formats in one application only (Jaakkola and Thalheim, 2021). Therefore, having systematic control over the (seemingly infinite) unstructured data sources could provide improvements in several aspects of Software Engineering (SE).

Recently, the trends of data management in applications cover the following aspects of data: the increasing amount of the data handled are either unstructured, or are based on the polymorphic data models. Data has a variety of sources, which are not under control of the application developers. This leads to the need for backward modeling. The application handles large masses of data due to a part of it having sources in legacy systems, and wide variety of changing (in time) open sources. The data flow is real time and a pipeline based. Traditional approach aims to develop applications for handling this complexity of data is based on by purpose solutions. However, there can be a lot of potential in the alternative approaches. In spite of the data complexity, it is possible to separate common aspects in the data management; this generic part (of the code) is application independent and reusable. The generic part is complemented by a context dependent specific part (of an application). Lastly, there may exist a part which is more or less beyond our control; it could be boilerplate or library code in the application, noise in the data or data formats, hacks to make the program to work, or simply has other features that are unknown to us. For the sake of completeness, it can be worthwhile to acknowledge this unspecific part in the architecture of an application.

Our approach to the challenge of data complexity is to provide an architecture of a framework, which separates the above mentioned three architectural parts. The generic and reusable part of the architecture is realized by Faucet-Sink-Drain Model based framework, which implements the application independent part of the system. The application specific part is connected to the generic part, and any unspecific features can be filled in by the unspecific part. The implementation of these three fragments is based on the principles of manageable (architectural) dependencies.

If data processing can be carried out more easily and more effectively, it will lead to improvement in the quality of the data. At the same time, improving the reusability of processed information can potentially save computational resources. Furthermore, development resources could be focused better if the overall complexity of a system can be simplified. It should be investigated how data *can* and *should* be managed and utilized by the software. This can be a multifaceted problem depending on the perspective of the potential user role. For example, a user might be interested in formulating a reliable algorithm to solve a specific problem. Another user might want to reuse and

extend this solution with their own (flavor of) data sources. When it is looked at from the software developers' point of view, they would like to have appropriate tools, interfaces, and means for creating a fully functional software application. In an ideal world, this can be done in a controllable and systematic way that can also avoid error-prone ad-hoc implementations.

This thesis approaches the challenge of data processing by proposing the design of a simple and generic model, usable by a wide range of applications. The main goals of the model are to standardize and to control the steps of data processing performed by a software application. In other words, it aims to provide a "mental model" for solving various data processing problems in software applications. Secondarily, the simplification by the model is based on the metaphor of a common and easily relatable household appliance.

The hypothesis is that the model enables improvements in the quality of the data processing as well as improved software development. The model proposed in this thesis is named Faucet-Sink-Drain Model. The model is realized as a prototype system where the principles of the model are applied and utilized. The chosen context for the prototype is data management. The prototype system is constructed in a way that the generic functionalities provided by the model can be separated from the main logic of the system.

The initial working name (in Finnish) of the model was playfully "viemärimalli," literally in English, a drain model. While this name is quite a verbatim characterization of the model, proper English names were needed. Therefore, the generic data processing model part is called Faucet-Sink-Drain Model while the data source management part is called Manageable Data Sources (MDS) component framework.

To completely define and understand Faucet-Sink-Drain Model requires that it be examined from multiple viewpoints. This thesis uses the following definitions of terms in the context of Faucet-Sink-Drain Model:

- As a model, it represents the conceptual idea of pipelined data processing. The model is defined using the metaphor of a streaming water apparatus which consists of faucets, sinks, and drains. In practice, it is a reusable generic model that can be adopted in applications and implemented, for example, as a software library or framework.
- As a design pattern, it represents a general and reusable solution on how the problem of data processing (and data management) can be approached in software applications.
- As a framework, it provides a ready-made implementation of the model and the basic functionalities that give guidance for the structure for a software application. A component framework is a distinct part of a framework, which has a clearly separated role and task: it can be assembled and reused by another system.

- As an architecture (or more specifically, an architecture fragment), it defines a logically cohesive system unit to solve various problems. The specific problem of this thesis concentrates on the effective and systematic management and processing of data sources. As such, this can be seen as an example of how Faucet-Sink-Drain Model (along with the MDS component framework) can be applied to solve a specific problem.

In essence, Faucet-Sink-Drain Model is a collection of various semantics assembled into a single term. Depending on the purpose, it is possible to choose the most functionally appropriate mode of operation from the defined semantics.

1.1 RESEARCH AREA

Data is the raw building block of all information. Data constitutes symbols that represent the properties of objects and events (Ackoff, 1989). Information is built upon data, knowledge is based on information, and wisdom requires knowledge (Ackoff, 1989; Rowley, 2007). Together they form a hierarchy, oftentimes called the Data-Information-Knowledge-Wisdom (DIKW) hierarchy. Literature has multiple definitions for each item in the DIKW hierarchy. For example, Rowley (2007) reviewed 16 papers to find definitions and the essence of the components in the DIKW hierarchy. The author concluded that the difference between data and information is that plain data has no meaning, as it is an unprocessed fact or observation. Information can be derived from data by organizing it or by including a purpose or context, making it meaningful, useful, and relevant. Furthermore, knowledge can be seen as a combination of information, understanding, capability, experience, skills, and values. (Rowley, 2007) Knowledge can be differentiated into explicit and tacit knowledge, where the former is something that can be recorded programmatically and the latter is an embedded part of the human mind (Nonaka and Takeuchi, 1995; Rowley, 2007).

In the context of SE, if one would like to get the benefits of the DIKW hierarchy, it needs to be programmed (or taught) to the software application. It is definitely not an easy task to carry out from a scratch, and presumptively there exist multiple approaches on the matter. The approach proposed in this thesis can be considered an examination of a tool for combining the strengths of the computer (e.g., good in repetitive work and memory) and human users (e.g., good in specialist knowledge). The design of a such system must also adhere to the study of software architectures for the production of high-quality and successful products (Taylor *et al.*, 2009).

Before the data can be turned into information, there are several questions that have to be resolved. In the case of software developers, they might be asking practical questions that require concrete answers, for example:

- Where does the data come from?
- Where is the data stored?
- How are the data and intermediary results cached and selected?
- Which kind of algorithms need to be used?
- How are the results presented?

The most obvious approach to solving these questions is to implement the software on an ad-hoc basis, but that could introduce unwanted side effects. For example, there might not be enough concern about how the achieved outcome could be utilized and reused in forthcoming systems or solutions. In other words, the new solutions tend to solve the questions again and again in their own unique way, which can accumulate with an unnecessary amount of resource consumption. Software standards such as ISO/IEC 25012:2008 (2008) define the qualities of the target data, but it appears that data management related features (for example, reuse and accountability of data) are not covered. The previously mentioned questions are referred to in a data processing model, which is presented in Section 3.1.2.

1.2 RESEARCH GOALS

The use of patterns, frameworks, models, and reusable code improves the quality of software (Nguyen, 2013; Greenfield and Short, 2003; Gamma *et al.*, 1994). For example, the Model-View-Controller (MVC) design pattern can be utilized to separate the application logic, data, and user interface into their own logical components (Buschmann *et al.*, 2013; Koskimies and Mikkonen, 2005). Software requirements and specifications, together with software standards are crucial in achieving the software that is fit for its intended purpose (Sommerville, 2016). Having well-defined requirements also help in other parts of development. The system will become easier to modify, maintain, and understand by dividing a complex system into smaller parts through the process of decomposition and modularization. These measures can positively affect the reuse of the source code. According to Sommerville (2016), the reuse landscape has an array of techniques available, where the selection of appropriate tools depends on the requirements for the system being developed. Therefore, we should strive to incorporate methods found in the software engineering literature (e.g., reuse and patterns) to fully exploit the potential of big data in data-intensive applications, to handle varying data processing and search-related requests efficiently.

There is a growing need for maintainable and systematic ways for managing the data produced, and subsequently, the information and knowledge that is derived from the data. Miksa *et al.* (2018) note that data has evolved from static documents into continuous streams of information flowing into Information Systems, and that users have to manage data efficiently to keep it safe and reusable. A general purpose

database may be an answer, but even then, the choice of an appropriate software tool depends on the circumstances as each piece of software is designed for a particular usage pattern (Kleppmann, 2017). For this dilemma, there are no comprehensive solutions available, which is an indication for a potential research gap addressed in this thesis.

Any such solution will demand the integration of several aspects. As a result, the Research Objective (RO) of this thesis is to identify a simple model of generic data processing that can be used for creating a systematic framework to process data requests. Further, the RO is divided into three Research Questions (RQs) which describe the deficiencies in the current situation.

RO There exists a generic model that enables the processing of data in any given software application.

- RQ #1 How to manage the data quality & data processing systematically?
- RQ #2 How to design and perform data processing cycle?
- RQ #3 How to comprise & execute the approach of generic model in a software application?

The presented RO and RQs are utilized to align the research with a SE focused solution for systematically managed generic data processing. The proposed solution is based on Faucet-Sink-Drain Model, which encapsulates a reusable design for data processing applications similar to what MVC is for User Interfaces (UIs). A prototype of a concrete framework has also been constructed. This construction is then described and scrutinized for the applicability of solving a problem.

1.3 RESEARCH METHODOLOGY

To conduct valid research, there should be a methodology that is suitable for the research discipline in question. The Design Science (DS) method popularized by Hevner *et al.* (2004) is a tool for conducting research in the context of IS, and presents a set of seven guidelines for DS research. These guidelines are useful for structuring and directing the research itself, and finally in the validation of the research approach once the research has been conducted.

Based on the prior work done on DS, Peffers *et al.* (2007) propose a Design Science Research Methodology (DSRM) for the production and presentation of DS research in IS. The DSRM includes the tools for carrying out the research that fulfills three objectives. These objectives support consistency with prior literature, provide a process model for conducting DS research, and a mental model for presenting and evaluating the research. (Peffers *et al.*, 2007) The methodology gives a suitable framework to follow for conducting rigorous IS research. Therefore, DSRM was selected as a research methodology in this study.

DSRM makes a synthesis of the common elements found in seven prior DS research studies. The result is a DSRM process model containing six stages of activities. The activities are described by drawing the necessary explanations from the studies, making it clear what the required inputs and outputs of the corresponding stage are. (Peffers *et al.*, 2007) The activities defined by Peffers *et al.* (2007) in their nominal order are as follows:

Activity 1 Problem Identification and Motivation

Activity 2 Objectives of the Solution

Activity 3 Design and Development

Activity 4 Demonstration

Activity 5 Evaluation

Activity 6 Communication

While the activities are listed in sequential order, it is not required to proceed in that order, and the process can be initiated "at almost any" of them (Peffers *et al.*, 2007). DSRM features an iterative process that allows a versatile approach to research. This enables returning to an earlier activity to conduct incrementally evaluative research activities, resembling a feedback method that eventually converges on a certain predetermined target. Identifying and adopting the research activities described by DSRM can provide a roadmap to follow during the research.

The initial activity can be identified from a prospective research entry point, and the entry points defined in DSRM correspond to a certain initial activity (Peffers *et al.*, 2007), which in this case is identified as the Activity 3 (Design & Development Centered Approach). The origin can be traced back to Publication V where the model of manageable data processing was presented for the first time. The problem identification and motivation (the RO in Section 1.2) appeared initially in Publication V but has been reviewed and slightly adjusted for the needs of this thesis. The earlier publications (i.e., Publication I–Publication IV) represent the prior research which supports the formation of the intermediate products of the research activities: inference, theory, knowledge of how to, knowledge of relevant metrics and analysis techniques, and knowledge of the disciplinary culture. An overview of the DSRM activities, the research specific steps taken during this thesis are depicted in Figure 1.2. Each activity is mapped to the publications of this compilation thesis.

This thesis also contributes to Activity 1, refining the contents of problem identification and motivation, bringing together the prior publications and the conducted research. The empirical knowledge was gained from the various prototypes described in Publications I–IV. As indicated above, those were produced prior to the identified Initial

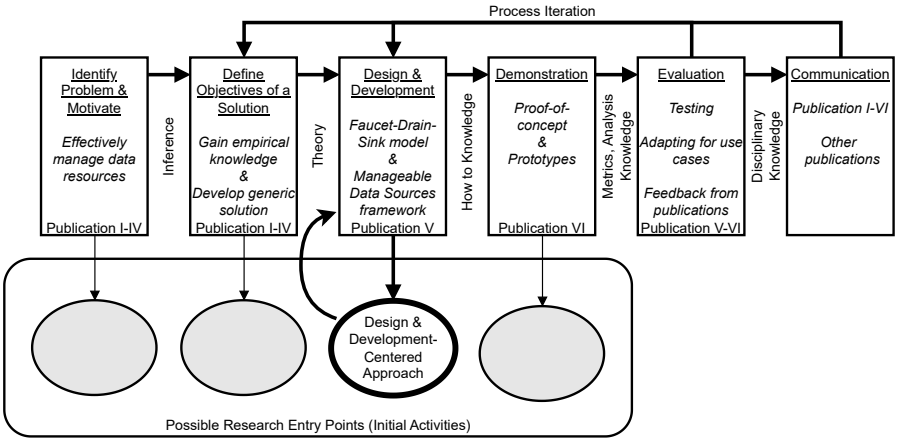


Figure 1.2: Applied DSRM process model. Adapted from Peffers *et al.* (2007).

Activity. Publications V and VI are results after it. All the intermediary publications contribute to Activity 6.

The outcome of the very first iteration of Activity 3 was an idea, which could be useful in general use data processing scenarios. This idea was refined iteratively through Activities 2–5 to find out what could be implemented and how to use this idea properly. Once the idea was found to be fruitful for further study, the original design of the model was published in Publication V. The general message from the peer review (prior the publication and at the conference) was that more progress in the study, and actual implementation and experimental results were expected. Hence, it was justified to continue the implementation of the system to gain the evidence of the applicability of the model. Ultimately, a prototype system based on the model was developed, demonstrated, and evaluated internally in several iterations. The objectives of the solution (i.e., the design and terminology of the model) were slightly adjusted once it was clear how to match the actual needs of the use cases. The intended feature set was also reviewed due to the limitations of available development resources. Practically the reference architecture was made less complex by dropping a few of the specific fields, and by applying more generic naming scheme. The next major step was the testing of the model in a proof-of-concept environment, published in Publication VI. Finally, the gathered research knowledge is summarized in this thesis.

LITERATURE REVIEW Both the DSRM and DS methods require the identification of problem relevance. This serves the need to show the importance of the research, gaining insight into the state of the art in the literature, as well as revealing a possible research gap.

To fulfill these needs, the literature review was conducted in two separate phases. The first phase can be seen as a very specific literature review. It was used to determine if there were any relevant or equivalent prior studies on the chosen thesis subject. The second phase was

a more generic review, in which the scope was zoomed out to find a broader spectrum of the relevant literature. More information about the literature review can be found in Section 2.1.

1.4 PUBLICATIONS

This section gives a synopsis of each publication included in this thesis, and identifies the independent contribution by the thesis author. The list of publications is given in the List of Publications on page xi, and the original publications can be found in Part II. The selected publications demonstrate the preceding iterations and prototype systems that were crucial in the conception of the model solution presented in this dissertation. The more detailed research contribution of the selected publications is given in Section 3.2.

PUBLICATION I "Utilizing Adaptive Software to Enhance Information Management."

Publication I deals with flexible and dynamic UIs for enhancing information transfer in mobile devices by presenting a tool for developing adaptive software. With the presented solution it is possible to improve information management in the software. The management, creation, and distribution of adaptive UIs, as well as access to the collected data, are evaluated with the adapter software approach. Adapter software is a specialized application that allows the usage of the created flexible application UI on any device by translating the downloaded UI definition file.

The contribution of the thesis author in this paper was the research on dynamic UIs, the implementation of the adapter software prototype by utilizing a cross-platform Qt programming framework, and in the documentation of the system. This paper can be seen as a continuation to Sillberg *et al.* (2011) from the information management perspective, where the thesis author strongly contributed to flexible UIs and data binding. The author wrote the main matter of the paper describing the dynamic UI system (i.e., sections three and four), prepared screen shots, figures, and formatting in the paper.

PUBLICATION II "Prototype System for Improving Manually Collected Data Quality."

Publication II presents a prototype system which is an attempt to improve the quality of manually conducted data collection. There are still legacy systems out there which cannot be replaced with fully automatic sensor systems, or where the desire is to keep them intact because ultimately they are working as intended. Nevertheless, it is important to gather the data that the systems generate. The prototype system helps the workload of maintenance staff by combining several cumbersome steps into one manageable step: the identification of the

data collection point, the collection of data by a pen and paper, importing the data from paper into a digital system, and finally processing the data. Additionally, the prototype can help in the elimination of human error, for example the mistyping of a number.

The author was responsible for implementing the client device application, and specifying the user-centered procedure of information collection facilitating the requirements of the maintenance staff. The data visualizations and processing functions were fully implemented by the author. Similarly, the author wrote the parts describing the overview of the prototype system and visualization of results (i.e., sections 3.0 and 3.2). Additional tasks included preparing screen shots, figures, and the paper formatting.

PUBLICATION III "Portable Sensor System for Reliable Condition Measurement."

Publication III extends the prototype system introduced in Publication II by replacing data collection with a commercially available automatic sensor device. In this study, more emphasis was placed on the visualization and post-processing of the collected data. The approach allows a higher sampling rate and provides scaling of the service by simply adding to the number of sensor devices. The device has a small internal memory, which has to be read periodically (e.g., once a month depending on the sampling rate) to avoid the loss of data. This also means that the approach does not fully support real-time monitoring needs, but allows considerably greater detail compared with manually collected data.

The contribution of the thesis author was the studying of the standards related to contactless identification systems, implementation of data import functions from third party sensors, and the realization of a floor plan view in the system. The system builds upon the results published in Publication II. The corresponding work in the publication is in presentation of the background and description of the implemented visualization UI (i.e., sections II and III.C). The author also prepared all screen shots, figures, and the paper formatting.

PUBLICATION IV "Web-User-Interface System Utilizing rHMEI and Open Data for a Water Quality Analyzer."

Publication IV introduces a prototype that is another application that relies on processing data and providing the user with a visualization. The prototype utilizes water sampling data from Finnish water bodies provided by the Finnish Environment Institute as Open Data. This means that anyone with an interest can gain access to Open Data. However, often the data provided by Open Data repositories requires specialized computer skills to obtain and analyze it. Thus, for non-specialist users there is a need for an easier way to interpret (or at minimum to inspect) the data. The prototype utilizes the River Heavy

Metal Evaluation Index (rHMEI) algorithm (Veesommai *et al.*, 2016) to translate the raw data into a more easily understandable representation of the measured heavy metal analytes in the water.

The contribution of the author was the implementation of the prototype application, i.e., the web UIs that collects the water quality data from the open data platform, and also the realization of the rHMEI in JavaScript programming language. The author also conducted the case study of the available water quality open data—ranging from 2010 until 2016—that gives an overview of the water quality (in terms of heavy metal analytes) in the municipality of Pori. The manuscript was principally written by the author while the role of co-authors was supportive in nature. They helped in reviewing and in giving the crucial insight on what topics should be addressed with, as well as confirming that the presented matters were factually correct. The author presented the work at the 27th International Conference on Information Modelling and Knowledge Bases (EJC 2017) in Krabi, Thailand, in June 2017. The paper was later published in *Information Modelling and Knowledge Bases XXIX* in 2018.

PUBLICATION V "Toward Manageable Data Sources."

Publication V marks the first appearance of the conceptual model of data processing and MDS. These concepts together lay out the foundation of this thesis. Further, the basic vocabulary and specification are laid out for the first iteration of the idea. In its simplicity, the data processing model consists of five basic components: (1) faucets, (2) streams, (3) sink, (4) sieves, and (5) drains. These and the other key topics are revisited in Chapter 4.

The author was wholly responsible for the contributions of this paper, and presented the work at the 28th International Conference on Information Modelling and Knowledge Bases (EJC 2018) in Riga, Latvia, in June 2018. The extended version of the paper was later published in *Information Modelling and Knowledge Bases XXX* in 2019.

PUBLICATION VI "Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data."

Publication VI presents a prototype system for gathering data generated by users' everyday actions by utilizing crowdsourcing and mobile phones. This system is intended for recording normalized accelerometer and location information provided by the various sensors of a mobile device. The collected data quickly builds up, and creates challenges such as how to effectively manage and process big data. The data produced is processed into five different shock levels ($L_0 \ldots L_4$) that can be visualized or further utilized by other systems. The shock level represents the severity of unevenness of the road, where shock level L_0 contains the lowest one-fifth and shock level L_4 contains the highest one-fifth of the normalized accelerometer amplitude data.

The author adapted the model introduced in Publication V together with the prototype system for road condition data collection. The data analysis and interpretation of the recorded shock levels were also carried out by the author. In the publication, the written work reflects in the parts that discuss analysis results, data processing and the integration of models in the prototype system (i.e., sections 3.2, 4, and 5.2). Activities also included being the corresponding author of the publication.

1.5 THESIS STRUCTURE

The dissertation is composed of two parts. The first part (Part I) includes the main content of the research. This part is divided into six chapters, followed by a bibliographical section. The chapters are structured in the following way:

Chapter 1 gives an introduction to the research topic of the thesis. The chapter presents the included publications, goals, and the chosen research methodology of the thesis.

Chapter 2 deals with the positioning of the thesis topic and presents the background of the research by examining the related studies revealed by means of a systematic mapping study. The methodology of the literature review is also addressed here. An examination of the applicable SE topics is made before drawing the conclusions of the background study.

Chapter 3 approaches the main research subject of the thesis by introducing the prior work of the thesis author. The key focus areas and selected SE topics supporting the forthcoming findings are described together with their linkage to the publications selected for this compilation thesis.

Chapter 4 presents the main results of the studies. The first section addresses the system model and architecture while in the second section the prospective use cases are studied. In the third section the prototype system based on the model is described, and in the last section the conclusions based on the findings are discussed.

Chapter 5 discusses the results of the studies by analyzing, evaluating, and validating the presented system and model. The chapter ends by outlining some future research topics.

Chapter 6 concludes the dissertation by summarizing the main results and findings of the thesis.

Finally, the second part (Part II) contains the original publications included in this compilation thesis. The rationale of the selected publications together with the author contribution is explained briefly in Section 1.4 and more thoroughly in Section 3.2.

2

BACKGROUND

This chapter outlines the related research and literature. The chapter is divided into four sections. The first section concentrates on the execution of the literature review. Insight into the literature discovered is provided in the second section. The third section is dedicated to the examination of the crucial software standards, models, and architectures. The last section summarizes the findings of the chapter.

2.1 IMPLEMENTATION OF LITERATURE REVIEW

The activities of the phases of the literature reviews conducted during this research are described in this section. Each subsection is detailing the findings and conclusions of a particular phase. While there are various literature review techniques available, systematic literature reviews (Kitchenham *et al.*, 2009; Babar and Zhang, 2009) have gained popularity in SE research. The opposite of a systematic literature review, informal ad-hoc surveys, are still preferred (Kitchenham *et al.*, 2009) by many researchers. Informal surveys are somewhat less laborious as they typically do not define the process of finding the literature, but undertaking a systematic literature review has its benefits in providing methodologically rigorous results (Kitchenham *et al.*, 2009). A systematic approach, such as a systematic mapping study, can support SE research by saving time and effort (Kitchenham *et al.*, 2011; Petersen *et al.*, 2015). The mapping study technique offers a less rigorous approach compared with a conventional systematic literature review (Kitchenham *et al.*, 2011). For example, Kitchenham *et al.*, 2011 mentions that the search strategy requirements are less demanding if only research trends are of interest. For the literature scope of this thesis, the mapping study technique was applied as the findings of novelty and research trends would be used as the basis for the decision of further research.

2.1.1 *Initial Review*

The first and the most specific literature review was conducted in November 2017. It was used to determine if there were any other relevant or equivalent prior studies on the chosen thesis subject based

on systematic literature review processes. The search was performed on three major scientific indexing search engines: IEEE Xplore, Scopus, and ACM Digital Library. Initially, the idea appeared simple enough, thus there was a genuine suspicion of the novelty of the idea. Regardless of concerns, it became clear that relevant keywords produced only a small amount of hits. Several variations were also tested independently to find better results. Finally, it was decided to utilize the following search term similarly on all indexes: "`(faucet OR drain OR sink) AND (data AND (conceptual OR model))`." The number of total hits (last column), studies related to computer science and SE (`Related Hits`), and studies that are both related and relevant (`Relevant Hits`) are shown in Table 2.1.

Table 2.1: Initial literature review.

Search Engine	Relevant Hits	Related Hits	Total Hits
ACM Digital Library			4
IEEE Xplore		3	11
Scopus		2	166

In the majority of the captured literature, the terms such as drain and sink were utilized in a different research field (for example, electronics or water resources), making those studies unrelated. Only five hits could be deemed related to the field of computer and information science. Finally, there were zero studies that were both related and relevant to the study area of the thesis. The conclusion of the initial review was that there were no exact matches and no research on exactly the same topic had been made previously. With this insight, further research on the idea was considered reasonable. The result was the formulation of the introductory publication, Publication V. In the making of the Publication V, the positioning of the research was refined, and the following related concepts were identified: architectures, big data, data management, design patterns, models, and quality of software. These topics were helpful in the discovery of the related research of the thesis.

2.1.2 *Generic Review*

The scarcity of results indicated that the keywords used in the first review phase were too constraining, and prompted a wider study. Using the feedback from the first phase, a significantly generalized concept was then formulated, resulting in the search term "`data management`". The search term was utilized similarly to the first phase. The second phase was performed in October 2019 while the proof-of-concept application was being tested with the data published in

Publication VI. The purpose of this review phase was also to make sure that the research was still aligned with the planned context and research goals. Therefore, a complete list of literature was not necessary, only the discovery of the most essential publications with enough penetration in the research area.

It was expected (and quickly confirmed) that this approach would generate a very large amount—hundreds of thousands, or even more—of search hits. Due to the huge amount of results we were at the mercy of the sorting algorithms of the selected search engines. In practice, only a small subset of documents could be taken for closer inspection. On the other hand, there are a few advantages to the approach as well. It reduces the impact of trending search terms and gives a relatively "fair chance" for any research to make an appearance. Another advantage, partly due to its random nature, is acquiring a wider and interdisciplinary point of view on the research topic. It helps to position the context of the thesis more accurately, and gives hints on where to look for supplementary literature. Finally, the method could reveal documents or studies that might have been missed otherwise (for example, due to poorly selected search terms). All the readily accessible and relevant documents obtained by this approach were then downloaded for closer inspection, and appropriately incorporated into the related studies. The number of direct citations were reassessed to capture the feasibility of the approach.

It was decided to include top hits from each search engine, and to classify the studies based on a brief subjective assessment of the provided document metadata (e.g., abstract, conference name, publication year, and so on). The search was conducted on six different search engines, yielding in 120 documents to be reviewed and classified. The selected search engines are listed in the header row of Table 2.2, and the qualitative and quantitative parameters are then listed on separated sections.

Let us start with the quantitative values (i.e., the bottom section). There were in total 13 duplicates, indicated by the `Duplicates` row. The duplicates are distributed among six documents: one document appeared three times and five documents appeared twice. Thus, there were in total 113 unique documents out of the 120 documents included. Duplicates are included in the total count of classified documents on each search engine. The effect of duplicates recounted in this manner is denoted by "(+n)" where n is the number of duplicated documents in the corresponding document class.

The `Unreached` row denotes the documents that were not accessible in electronic format with a reasonable amount of effort. `Total hits` is the number of search hits reported by the search engine. The number of search hits reveals one of the weakest points of the random hit search method; at best, only 0.2 percent of the results were covered by selecting the top 20 results. Thus, there is a high chance that relevant

Table 2.2: Generic literature review.

Parameter	ACM[1]	Google[2]	IEEE[3]	SD[4]	Sco.[5]	WoS[6]
Abundant		2	3(+1)	1	5(+1)	3(+1)
Adequate		5	3	6	5	4
Moderate	4	4	10(+2)	11(+1)	8(+2)	4(+3)
Duplicates	1		3	1	4	4
Unreached	1	3				1
Total Hits	22764	267000	9906	55667	36839	19035
Year (avg)	2011.5	2002.4	2012.4	2018.6	2011.4	2010.6
Year (std)	4.17	5.65	5.62	0.86	4.14	7.61

[1] ACM Digital Library [2] Google Scholar [3] IEEE Xplore
[4] ScienceDirect [5] Scopus [6] Web of Science

results will be missed in this approach. However, the exercise produced a good cross-section of interdisciplinary fields and quite a wide time range coverage. Useful follow-up references that appeared in some of the documents also helped in searching for other publications. These points positively contributed to gaining a wide perspective and a good overview of the subject of data management.

The last two rows indicate the publication year of the papers in the respective set; the first is the arithmetic mean of all documents, and the second is the standard deviation of the population. ScienceDirect produced the most recent documents (also with the smallest standard deviation) while Google Scholar produced the oldest documents. Google Scholar also returned the most search hits, having almost twice as many hits as the other search engines combined.

Returning to the qualitative values, documents were basically classified in four different categories based on a subjective assessment of the document metadata for conducting a closer inspection of the most related papers. A document that was immediately seen as out of scope was discarded (for example, a paper discussing medical sciences). The lowest document class (`Moderate`) was gained if the paper displayed any relation, but the connection was not strong enough, or if there was a chance that the paper could be deemed off-topic. A paper which included interesting and relevant references was placed in `Moderate` class. `Adequate` papers had a more relevant connection and/or dealt with related topics more clearly. The best papers (`Abundant`) were immediately seen as being related and highly relevant.

After the initial classification had been conducted, the remaining papers were then reassessed by inspecting their full contents. The final document class could then be readjusted based on the content. It transpired that all (12) `Abundant` papers were cited, whereas three (3)

`Adequate` and ten (10) `Moderate` papers were included. The `Abundant` papers were useful in describing similar studies and giving further insight into the study area. The papers in the lower classes were useful in the setting of boundaries (i.e., scoping) of the study area and in showing slightly different aspects of the topic of data management.

2.2 RELATED STUDIES

The related studies that were discovered in the conducted literature reviews as well as other topics that emerged are explained in this section. Each study presented in this section appears under a certain category (i.e., subsection) which covers the main characterization of the topic. Categories were selected on the basis of the identification conducted in the aforementioned literature reviews. A study may belong to several categories, but it is discussed once under the most relevant topic.

2.2.1 *Data Management*

The lifecycle of data comprises many activities, and using the data purposefully as a valuable resource requires knowledge about data management. Therefore, the focus of this subsection is on data management in general, and related issues and requirements.

In the literature, the meaning of the term data management varies depending on the disciplinary field, for example in telecommunications (Franklin, 2011; Bisdikian *et al.*, 2011) it may refer to managing the amount of transferred data. There are also cases where data management is mentioned together with optimization issues, e.g., regarding the limited computational resources of computer hardware (Cheng *et al.*, 2017), scalable systems and services (e.g., Agrawal *et al.*, 2010; Abadi *et al.*, 2007), as well as challenges with big data processing (Khan *et al.*, 2018; Emara and Huang, 2019; Kantere, 2014). Studies collecting scientific data repositories, such as clinical trials (Gazali *et al.*, 2017; Hossmann *et al.*, 2017; Du *et al.*, 2016) and others (Pophal, 2019; Dong *et al.*, 2010), refer to managing electronic research data in general, for example in terms of accessibility, shareability, storability, and distribution. Engineering, manufacturing, and business related fields tend to talk about managing the product configuration (e.g., Li *et al.*, 2020; Chen and Tsao, 1998; Qiao and Liu, 2009).

There are research studies and guidelines covering the needs of the data management of research data (e.g., Finnish Social Science Data Archive, 2015; Bellgard, 2020). Data management for research institutions, such as universities, can be challenging not only in terms of actually handling the data but also in organizational policies governing Research Data Management (RDM) (Bellgard, 2020). Planning of data management has also been studied, for example by Miksa *et al.*

(2018) and Vieira *et al.* (2014). When looking for standards, instructions, and test procedures specific to environmental data management software, Monitoring Certification Scheme (MCERTS) based publications by Lloyd (2007) and Environment Agency UK (2017) might be useful.

Studies about data maturity are made under the term Data Management Maturity (DMM). Baolong *et al.* (2018) have performed a survey of several methods, e.g., DMM by CMMI Institute (2019) and the data governance capability maturity model from IBM (Soares, 2010). They also provide a brief comparison and typical use scenarios of them, including their own capability based DMM, the Data Capability Maturity Model. DMM gives valuable guidance on the quality, governance, architecture, and strategy aspects of data and data management.

Many solutions can be characterized as Data Management Systems (DMSs). Systems such as Apache Spark (The Apache Software Foundation, 2018), and the Microblogs Data Management System (MDMS) by Magdy and Mokbel (2015) are used to overcome the limitations observed in traditional Database Management Systems (DBMSs). The MDMS resembles a Structured Query Language (SQL) database backend, but is tuned for the high input data rated social media platforms.

Wilkinson *et al.* (2016) concentrate on the management and stewardship of scientific data. It is mentioned that "good data management is not a goal in itself, but rather is the key conduit leading to knowledge and innovation." They also consider that there is a great need for improvement of the infrastructure that supports the reuse of scholarly data. They describe and explain a guide to Findable, Accessible, Interoperable, Reusable (FAIR) principles. In general, the FAIR guideline is an implementation agnostic, and does not specify a standard or suggest how to implement such a system. They present six examples of systems which are FAIR compliant (or partially compliant), and their level of "FAIRness" aspects. The FAIRness of the Faucet-Sink-Drain Model is not evaluated, but it is important to consider the principles.

All things considered, data management as such is a widely researched topic that is in great demand. The common denominator for most cases is that there is a need for handling the data that is being generated by various activities (by sensors, computers, humans, etc.), and that the issues are resolved by implementing case-specific solutions. This indicates that there are no comprehensive general purpose data management solutions available. Ridley and Stoker (2004) describe—based on Thuraisingham (1998)—many important points, qualities, and advantages of a good DMS. They conclude that a good DMS allows data to be used quickly, efficiently, and reliably. Data usage at its maximum potential can also mean increased overhead on data import and cleaning up, but the "rewards can be immense" by taking the trouble and going through the process.

Finally, in the context of SE, it should be avoided to implement DMS solutions from the scratch on a case-by-case basis because repetition

lowers the productivity. Building upon the tradition of SE design patterns, and through the improved data reusability, the data processing will become more efficient. As such, various applications may now utilize the same solutions and processes, saving time and resources.

2.2.2 *Similar Approaches*

This subsection comprises applications and models that have influenced, have similar characteristics, or possess other important features that are positioned in parallel with the topic of the thesis. Even though the Faucet-Sink-Drain Model has not yet been explained in detail, the similarities, differences, and strengths across the found approaches will be summarized at the end of this subsection.

Xue-rong *et al.* (2009) study Universal Simulation Data Management System (USDMS) and the data ownership management in it utilizing the Basic Data Object metamodel. The metamodel is used to organize what is related to the data, and it is composed of three components: 1) basic information used for natural language identification, 2) property data that is statically defined for each simulation task, and 3) status data which contains the dynamic run-time information of the task. Xue-rong *et al.* (2009) also introduce an implementation of USDMS in relational DBMS. The interesting in this study is the usage of metadata together with the proposed metamodel.

Kantere (2015) introduces the interesting concept of Datom, a modular data management solution. Datom, or data management atom, encapsulates generic data management provisions in terms of data workload and computing resources. The modularity of the approach suggests that it is capable of generic data processing. The design goal of Datom is to provide a tool for express problem constraints, assumptions, and requirements in a unified and systematic manner.

A closer look at Datom is shown in Figure 2.1, which depicts the structure of a single Datom (a), and a collection of more complex sets of Datoms (b). It can be seen that Datom consists of three elements: data, workload, and computing resources. The author further discusses the usage of Datoms for data management entities in Kantere (2016).

A generic and distributed data management approach, Generic Data Management System Architecture (GDMSA), by Plantikow *et al.* (2009) deals with the challenges of community grid data management to better understand how a one-size-fits-all data management system could be built. Community grids are related to grid computing infrastructures. The proposed architecture addresses the complex distributed data management issues by separating and handling the different concerns using physical, logical, and query-based addressing. The concept for the next generation data management includes several interesting points such as generic data access Application Programming Interface (API), which they believe would be beneficial for the

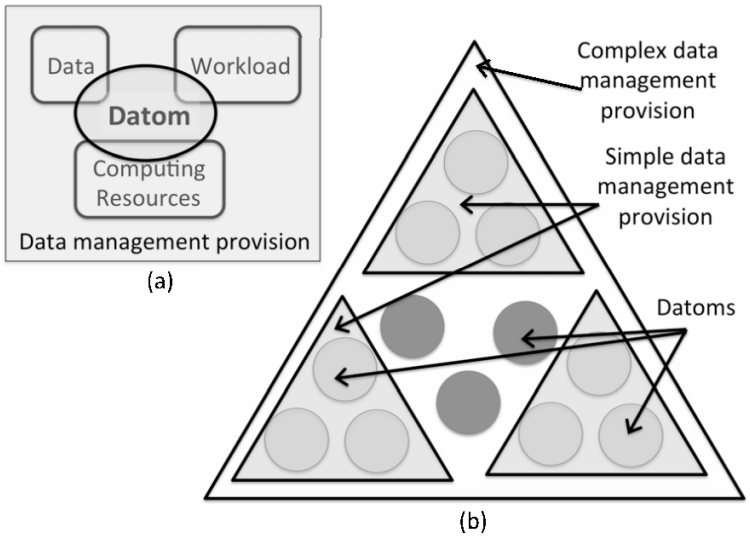


Figure 2.1: Overview of Datoms and data management provisions. (Kantere, 2015)

application developers of community grids. Similar to GDMSA, a cloud-based data management framework, Flexible Robust Intelligent Elastic DAta (FRIEDA), by Ghoshal and Ramakrishnan (2012), attempts to overcome the data access issues of transient cloud environments by providing application-specific mechanisms for managing the transfer and storage of data.

The Data Warehouse (DW) design was initially reviewed in Publication V, thus only a brief summary and comparison is included in this thesis. The purpose of a DW is to provide an architectural model to support the decision-making activities of an organization. It is achieved by extracting and storing variable data in a uniform format in a centralized database, which provides tools for querying, reporting, and information analysis. The definition by the author of DW (Inmon, 1992) says that "[a] data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions."

Data lakes are a newer, unstructured variant of data repositories, that differ from DWs by having more flexibility and fewer constraints. Data lakes are characterized by being like a "body of water in its natural state," whereas DW is more like purified, bottled water ready for utilization. Their different approaches to data storage also mean that they have different tools for accessing data. Whereas DW is more business-oriented with its prestructured data use cases, data lakes are better suited for scientists and experts that embrace exploratory examination of the data by "diving in" to the lake. (Campbell, 2015; Deloitte Consulting LLP, 2018)

Carney *et al.* (2002) present a system called Aurora, which is aimed at real-time operations and offers a stream-oriented set of operators. Their research concentrates on a DBMS model for monitoring applica-

tions, and they list five key points (data from external sources; tracking of recent historical data; trigger-oriented operation; data storage issues; real-time requirements) that make the traditional DBMSs difficult to integrate with monitoring applications. Their Aurora system is based on stream-oriented input data sources that are processed multiple times on their way to the output stream stage (i.e., before being presented to data requesting applications). There may be several queries to control the output stages and it can maintain historical storage.

MongoDB is an open source document-oriented NoSQL DBMS which features an aggregation pipeline framework based on the concept of a data processing pipeline (MongoDB, Inc, 2019). An example of defining an aggregation pipeline is shown in Listing 2.1, containing a scripted pipeline of a two-stage operation. The first stage selects the documents which match the given `status`, "A." The second stage groups together the documents by their unique identifier, "$cust_id," calculates the sum of the field "$amount," and places the results (`_id` and `total`) into new documents. (MongoDB, Inc, 2019)

Listing 2.1: MongoDB Aggregation pipeline.

```
db.orders.aggregate([
  { $match: { status: "A" } },
  { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
])
```

There are other scalable database- and analytics-oriented tools such as Elasticsearch (Elasticsearch B.V., 2021) which is based on Apache Lucene library (The Apache Software Foundation, 2021b). It is a search and analytics engine for all kinds of documents. However, it does not handle data management by itself. Apache Spark (The Apache Software Foundation, 2018) is also quite similar in this regard, and can be configured to provide a unified analytics engine for large-scale data processing. The Apache Flink (The Apache Software Foundation, 2021a) is an interesting concept of a unified stream-processing and batch-processing framework. For example, it employs data pipelines that can improve the latency of moving data and can also continuously consume and emit data. Flink can be used to develop many different types of applications due to its extensive features set. Flink does not explicitly point out its data management features, but those could be incorporated by using the package ecosystem for third-party projects.

SUMMARY The reviewed tools are employing interesting concepts, but among the most interesting are the variants of pipelining and streaming features which do appear most often. Kleppmann (2017) considers that stream processors are suitable tools for data integration in data-intensive applications because it makes sure that data ends up in the right form in all the right places. However, while the discov-

ered tools solve problems in their respective domains, they are not a comprehensive solution for reusable general purpose applications.

The Aurora system has similar idioms to Faucet-Sink-Drain Model, such as streams and built-in operators, but the main difference is that Aurora is designed as a stand-alone DBMS program, whereas Faucet-Sink-Drain Model and its features can be integrated directly into the application. The integration is one step further to a comprehensive general purpose data management solution. Aurora is also more output-oriented, whereas Faucet-Sink-Drain Model focuses on management of the sources and controlling the accountability and reuse of information.

The aggregation pipeline of MongoDB has similarities with the Faucet-Sink-Drain Model in the part where the documents can be fed into a multi-stage pipeline to be aggregated and grouped into more interesting results. There is one stage in the aggregation pipeline which corresponds to the filtering and processing (performed by a data sieve) in Faucet-Sink-Drain Model.The advantage provided by Faucet-Sink-Drain Model is that it will be able to track and manage the complete path of the steps taken in the processing of the results.

The similarity of Datom and Faucet-Sink-Drain Model is high in terms of offering a unified, systematical, and reusable solution. However, the abstraction level of Datom is at a higher level. The data management provision of Datom consists of three sections: data, workload, and computing resources, while Faucet-Sink-Drain Model concentrates on the data part only. Faucet-Sink-Drain Model describes how the data sources should be utilized by the software (Publication V). Nevertheless, the topics and arguments presented in Kantere (2015) are well aligned with those of Faucet-Sink-Drain Model, which make it reasonable to believe that Faucet-Sink-Drain Model is compatible with Datom.

Faucet-Sink-Drain Model resembles the concept of DW in several ways but the key difference is that the input data can be imported with more flexibility and with less prior design effort. For example, the data produced by different sources are imported to DW by utilizing extraction-transformation-loading processes before the creation of smaller logical sections called data marts. Data marts, which are comparable to Data Streams, are more static and unchanging representations of an identified business problem. Defining streams in Faucet-Sink-Drain Model is more flexible as the logical size of a stream is smaller than that of a data mart. The downside of this is that the number of required streams will be much greater, as less data processing is performed per single stream.

The Basic Data Object metamodel that was introduced in USDMS resembles the faucets, streams, and data defined in Faucet-Sink-Drain Model. The idea of data lakes provides an unstructured data processing approach compared with the DW, but both terms have been

criticized for being too fluctuating and vague. Apache Flink offers an interesting architecture in pipelined data processing which could offer synergy benefits in the future development of Faucet-Sink-Drain Model.

2.3 SOFTWARE STANDARDS AND BEST PRACTICES

SE and the development of software relies strongly on guidelines given by various standards and best practices gained from well-tried architectures, models, software frameworks, and design patterns. This section introduces the selected standards and other topics that support the effort to attain the goal of this thesis.

2.3.1 *Standards*

The literature review indicated that there are several major entities that provide standards in the field of data management and data quality. Data quality and software standards are mainly covered by publications from International Organization for Standardization (ISO) while data management standards include a few other players. In the following, a selection of the most relevant standards will be examined.

DATA QUALITY The ISO/IEC 2501n series, or *Quality Model Division*, is focused on a quality model of both software and data. It belongs to the family of Software product Quality Requirements and Evaluation (SQuaRE). SQuaRE is organized as follows: standards numbered 25000–25049 form the five core divisions of the series. The following documents (25050–25099) are reserved for various extensions such as requirements for the quality of commercial off-the-shelf software (ISO/IEC 25010:2011, 2011). To set the baseline for software quality in this study, the ISO/IEC 25010:2011 (2011) defines quality as "a degree to which a software product satisfies stated and implied needs when used under specified conditions." ISO/IEC 25010:2011 replaced ISO/IEC 9126-1:2001 (Software engineering – Product quality) which was originally issued in December 1999.

Quality models are useful for a wide range of users and benefit product development activities such as the identification of software and system requirements and design objectives (ISO/IEC 25010:2011, 2011). The model for data quality from ISO/IEC 25012:2008 (2008) is complementary to the product quality model defined in ISO/IEC 25010:2011. The product quality model defines eight quality characteristics: *Functional suitability*, *Performance efficiency*, *Compatibility*, *Usability*, *Reliability*, *Security*, *Maintainability*, and *Portability* (ISO/IEC 25010:2011, 2011). Each characteristic is broken down into several subcharacteristics, e.g., completeness, capacity, accessibility, availability, modularity, reusability, to name a few (ISO/IEC 25010:2011, 2011;

ISO/IEC 25012:2008, 2008). A selection of practical characteristics for the thesis topic will be discussed in Chapter 3.

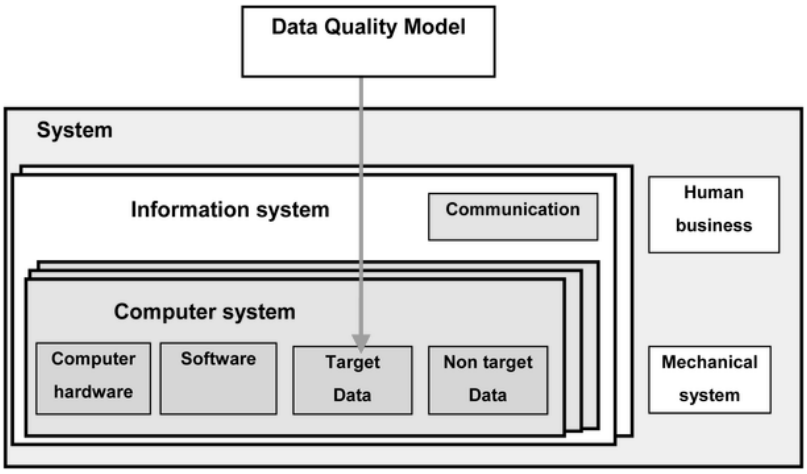


Figure 2.2: ISO/IEC 25012 Target domain. (ISO/IEC 25012:2008, 2008)

The general data quality model defines a structured format within a computer system, and specifically the portion of *Target Data*, as seen in Figure 2.2. The standard specifies that "target data are those that the organization decides to analyze and validate through the model." Thus, there are portions of data which are within the scope of the standard but also there may be some non-target data to be left out from the standardization process by a conscious choice. The structure of the `System` can consist of multiple `Information systems` which in turn can include one or more `Computer systems`. (ISO/IEC 25012:2008, 2008) The topic of this thesis will primarily refer to inside a computer system, specifically in the `Software` section, by making use of the data quality requirements specified for the target data.

Rafique *et al.* (2012) provide an extensive evaluation of the ISO/IEC 25012:2008 data quality model in terms of information quality and learnability. They go one step further from the quality of data, and define an information quality model "InQ" with the use of the ISO/IEC data quality model. They also want to make a clear distinction between data quality and information quality by stating "[w]hen data are put into a context and combined within some structure, information emerges."

BIG DATA To handle the extensive amount of data, the ISO/IEC standardization organization has taken the step of defining the big data paradigm, for example in the following standards and technical reports ISO/IEC 20546:2019 (2019), ISO/IEC TR 20547-2:2018 (2018), and ISO/IEC 20547-3:2020 (2020). The first (ISO/IEC 20546:2019) aims to set the definitions and the vocabulary needed to form a common understanding for big data, as the term has been widely overloaded with many kinds of scenarios (which may or may not be the actual

manifestations of big data systems). The following report consists of five parts which are useful for a deeper understanding about big data. For example, part 2 provides use cases and derived requirements, whereas part 3 introduces the big data reference architecture. Many of the parts in ISO/IEC 20547 were still under development at the time of writing this thesis.

DATA MANAGEMENT MCERTS is a standard for data management applications software to deliver quality in environmental measurements (Environment Agency UK, 2017; Lloyd, 2007). The standard is organized into three parts: part A, which is the foundational layer covering generic software quality; part B, which is focused on the performance of generic data management; and finally part C, covering additional performance aspects for specific environmental measurement applications. The principal author of the standard goes through one of the initial versions of the standard in Lloyd (2007). As an example, the standard has identified continuous emission monitoring systems in its specialized standards. The thorough and check-list style of the standard offers a rigid and potentially useful guideline for any software, not just environmental sensing applications.

DMM was briefly mentioned above by Baolong *et al.* (2018), who conducted an analysis on a few capability maturity models as well as one of their own. DMM is one way of improving the quality of data, mainly through the discipline of treating the data as an asset of an enterprise (Soares, 2010). While the organizational process and the governance of data are not the main topics of this thesis, DMM is a way to emphasize the importance of data. CMMI Institute (2019) states that, with their guidance, the organization can identify strengths and gaps in their data assets to improve their business performance. With the above in mind, and the fact that there are standards for almost each and every case available from many sources, it is crucial to select standards that can be applied to specific needs, environments, and resources (Horch, 2003).

2.3.2 *Best Practices*

In SE the more lightweight counterparts to software standards, such as design patterns and models, can form the best practices that are useful in cases where the absolute consistency and rigidity of standards are deemed inappropriate (Horch, 2003). The goal of utilizing the best practices is to help to make good choices when designing software, and to achieve better and robust solutions. The industry-proven design patterns, by Gamma *et al.* (1994), are intended to "solve specific design problems, and make object-oriented designs more flexible, elegant, and ultimately reusable." The reusability of a software artifact is the key to reducing repetition in software development.

In object-oriented programming and design, any well-structured architecture is full of design patterns (Gamma *et al.*, 1994; Sommerville, 2016). Architectures are a tool for understanding and to build a bigger and bigger entirety of systems through the increased level of abstraction. Additionally, the long-term trend of SE is to increase the level of abstraction of software descriptions, and to produce functional systems from even higher level representations (Koskimies and Mikkonen, 2005). ISO/IEC 42010:2011 (2011) defines the architecture design, descriptions, as well as conceptual foundations among other factors.

Software architecture can be seen as the base structure of a system, which defines the rules and boundaries for conducting more detailed design and implementation. On the one hand, it takes away some creativity, but on the other, it makes the design of software easier and faster to build (Gamma *et al.*, 1994). A software architecture is a collection presentation that can describe the architecture from several viewpoints, by different stakeholders or levels of abstraction. One of the most important factors is to keep the architecture well documented, for example, by following the guidelines laid out by Clements *et al.* (2010). The guidelines set the rules for common language so that the reader of the documentation can clearly follow it.

A software framework is a collection of components, and/or interfaces that can implement a base functionality of a specific need, and is often used to create product-line architectures. Software frameworks are designed to be reusable from the architecture level. To attain reusability, the frameworks employ a technique called the inversion of control. In addition, frameworks have known gaps (i.e., specialization interfaces) in their implementation, allowing for the specialization of the framework. The desired application artifact is achieved by filling in the gaps with new code that implements the required functionality. Design of a software framework is not an easy task, and the use of a framework almost always incurs some trade-offs and risks. The controllability of a framework can be managed with proper requirements, definition, analysis, structuring, implementation, and testing. (Gamma *et al.*, 1994; Koskimies and Mikkonen, 2005)

Adopting these good software design choices from the beginning, the maintenance of a software program is easier and the chances for problems arising in the future are lower. It can be assumed that it is worth implementing such measures in applications that manage and process very large amounts of variable data. If the hypothesis is correct, it is possible to create a new (data) design pattern style for a new (data) environment, an experimentation that is incorporated in this thesis.

2.4 CONCLUSION OF THE BACKGROUND STUDY

The chapter has addressed the literature background of the thesis and described how the reviews of the literature were conducted. The initial findings of the first review indicated that no exactly similar research had been carried out. Thus, the second review phase was conducted with wider scope.

A further review revealed that data management has been studied widely and in several interdisciplinary studies, i.e., not only in the field of SE but also in medical, engineering, manufacturing, and business sciences. A common factor for many studies on software applications is that there is a certain need—simple or complex—for handling the data that is generated by various activities. This need is then resolved by implementing software which is a highly specialized solution. While the software may utilize reusable software components, following the software standards and best practices, it appears that often the data management logic of the software is being "invented" and implemented again and again for each application. This difficulty could be resolved more efficiently by simplifying and generalizing data management processes.

It was mentioned previously in Section 1.2 that there is a growing need for maintainable and systematic ways for managing the produced data, and also an appropriate software tool to fully take advantage of data-intensive applications. After reviewing the related research, it can be concluded that there indeed is a research gap in having a comprehensive solution available.

Therefore, the model introduced by this thesis can fill in the gap, in practice, by following the SE tradition of design patterns. Using the tools provided by the model guides the way to replace those previously highly specialized data utilization logic with generalized and reusable counterparts and processes. The resulting model and its prototype framework will increase quality and reuse in data-intensive applications.

3

TOWARD GENERIC DATA MANAGEMENT

This chapter explains the selected work of the thesis author, and the steps to collect the empirical background which has led to the realization of the generic data management model and the framework utilizing it. This dissertation is a compilation-type thesis which includes the selected articles that directly support certain parts of the findings of the aforementioned model. These publications were briefly introduced previously in Section 1.4. The work with multiple prototypes, pilot systems, and other software project work items have greatly contributed to the forming of knowledge and to understanding the phenomena occurring in the processing of data in information systems. With this knowledge, the integration of different software systems raises the importance of models and architectures.

3.1 IN SEARCH OF DATA MANAGEMENT

It can be frustrating to adopt data sets prepared by others—even those made by ourselves. It always takes time to get familiar with the data and other documentation. Even with good instructions, it is almost certain that the use of case-specific requirements necessitates modifications to the initialization code or the data set itself. Even though the task might be easy and fast, it can begin to feel repetitive and cumbersome after doing it for the n-th time. It certainly would be useful to have a system that can generate and maintain "automated recipes" for reproducible and reliable data management solutions.

The amount of data available—more or less Open Data—has given plenty of opportunities for combining the data intelligently and providing new ways to contribute to society. However, retrieving and processing the available data requires those external systems to be integrated. Practically, their documentation, interfaces, and architectures need to be learned beforehand in order to be able to progress in our own goals. Formalizing the process of problem solving with models and architectural views would be useful for software development, but it can also be reflected in the completed artifacts for the benefit of the end users.

In one of the author's joint publications (Sillberg *et al.*, 2013) it was stated that "[t]he amount of electronically stored and unclassified data

is growing continuously, resulting in the need to find smart ways to manage all this data." Although this article was published several years ago (in 2013), the statement still appears to be valid. It just might require the following clarification: the paper was extensively focused on the end users of the software, but in fact, there should be similar management mechanisms and tools for *everyone*. Thus, the needs of software developers, stakeholders, non-specialists in information and communications technologies, third parties, and so on, should be equally important.

3.1.1 *Quality Aspects*

Despite ISO/IEC 25010:2011 (2011), which states that data quality is a significant part of the quality of a system, some interpretations appear to take it for granted. It may lead to the assumption that data quality simply appears together with software quality. Efforts must be made to make data reusable, just as it is in the case of software, or a system. Due to the ever-increasing amount of data—or big data—the quality and management aspects should receive more attention from practitioners. The starting point is the product quality model in Figure 3.1 defined by the standard. By extending the quality model (for example, in the same manner as was done by Rafique *et al.*, 2012), it should be possible to utilize these categories and/or their sub-characteristics in defining extensions for data management and data quality.

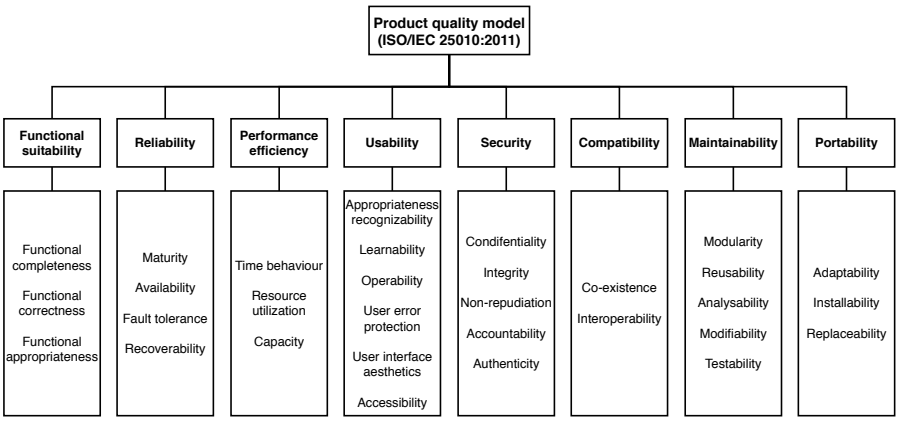


Figure 3.1: Product quality model. (ISO/IEC 25010:2011, 2011)

However, a perfect solution might not be realistically achievable. Improvement of quality parameters is also a matter of finding an optimal balance with the workload; thus improving one parameter may impair another parameter. For example, increased maintainability might adversely affect performance efficiency and thus increase demands for computational capacity. Of course, if we are given an unlimited amount of development resources and time, almost everything is possible, but this is not feasible in typical business-driven contexts.

Therefore, some trade-offs must be made in order to properly focus on the selected key areas that we are interested in. Selecting parameters that support each other will make it easier to achieve the goal.

Based on the experience in the development of complex (data management) systems (i.e., Publication I–Publication IV), we have observed similar and recurring issues which are related to the quality of data and software. In the scope of this thesis, the goal is to find a solution that can improve quality aspects with a reasonable amount of trade-offs. Furthermore, the selection of quality parameters is based on the observations. For this reason, an extension of the quality model dealing with data quality and data management was considered. It transpired that there were a few characteristics in the product quality model (ISO/IEC 25010:2011, 2011) which could be repurposed for this need. As a result, the chosen characteristics would be useful in drafting the *data quality and data management* extension.

The following list contains the quality parameters that are appropriately balanced for the above mentioned goal. Additionally, the Faucet-Sink-Drain Model introduced in this thesis should—and will—be able to provide a solution that supports these quality parameters. Each of the listed characteristics is followed by a short textual description of how the characteristic is related to data quality and data management. After the description, the most closely matching category/sub-characteristic with the informative description from ISO/IEC 25010:2011 is given as a reference (preceded by • and *written in italics*):

Accountability deals with the traceability of data, i.e., the paper trail of operations performed with the data and results. It can explain precisely the different steps, and can help in recreating the results. The tracing can also be used in the fine-tuning of algorithms as the changes between different experiments can be compared.
• *Security.Accountability: The degree to which the actions of an entity can be traced uniquely to the entity.*

Compatibility and Interoperability deal with the ability to use the data on any generic system or software. They aim to minimize the efforts required in preparing the data for use.
• *Compatibility: The degree to which a product, system, or component can exchange information with other products, systems, or components, and/or perform its required functions, while sharing the same hardware or software environment.*
• *Compatibility.Interoperability: The degree to which two or more systems, products, or components can exchange information and use the information that has been exchanged.*

Reusability and Reproducibility deal with the manageable reuse and data reprocessing in cases where results are needed again. Being able to obtain reproducible results consistently is important if the determinism of the system is required.
• *Maintainability.Reusability: The degree to which an asset can be used in more than one system, or in building other assets.*

Reliability and Availability deal with the ability to obtain proper and current data as well as having the capability of retrieving the previously stored data and/or results.
• *Reliability: The degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time.*
• *Reliability.Availability: The degree to which a system, product, or component is operational and accessible when required for use.*

It should be noted that the entries use a similar sub-characteristic naming convention and sometimes overlap with existing characteristics that can already be found in ISO/IEC 25010:2011. For this reason, more careful naming schemes and more formalized textual descriptions must be adopted to make the new data quality and data management category more suitable for incorporation into the standard extensions.

3.1.2 *System Models*

Even a simple system that accomplishes one task may consist of multiple software components and artifacts, which in turn may implement different models and design choices in order to achieve their goals. System modeling is the key to simplifying the design process, but it requires an increase in the level of abstraction, and thus a reduction in the amount of details. Using several layers of abstraction allows a finer grained separation of concerns in the underlying subsystems, and can offer multiple viewpoints regarding the structure of the system. A model can also be seen as the blueprint of the foundation of the system in question. These foundations can be used to describe the fundamental rules that need to be complied with when developing software.

Therefore, the data flow of a system should be modeled. In Figure 3.2, a model of a fairly typical information system conducting data gathering, processing, and visualization is shown. The presented prototype system consists of three different subsystems (i.e., devices): a mobile phone for data gathering, cloud service for data processing, and a web-browser client to facilitate the visualization of processed data (to give the user a way to consume the information that is generated).

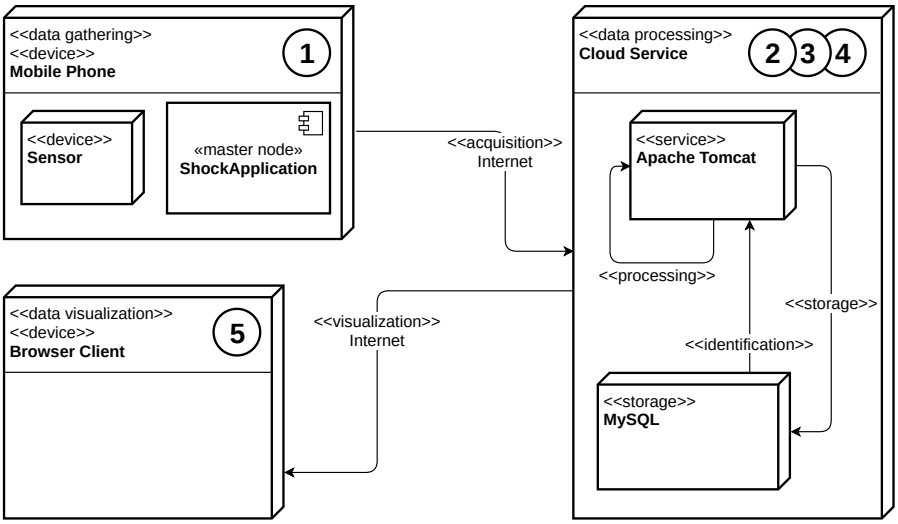


Figure 3.2: Deployment of prototype system. (Publication VI)

Each subsystem is responsible for at least one high level task related to data processing. In Publication VI these tasks were defined as: 1) acquisition, 2) storage, 3) identification and filtering, 4) processing, and finally 5) visualization. These tasks also combine the key focus areas of the previous study by the author of this thesis. The focus areas are as follows:

1. Acquisition Data gathering is performed by any capable device (e.g., a mobile phone). The device may operate as a combined sensor-master node if it is also capable of communicating the data to a remote service.

2. Storage The cloud service receives and parses the data sent by the master node. The parsing of the data needs to be done before the received data can be fully utilized. Once finished, the service will be able to store the data in a format that it recognizes.

3. Identification and Filtering The data will be identified and filtered when the service receives a request on its interface. Data selection is based on the criteria that were passed in the parameters of the request.

4. Processing The selected data is processed further by the rules (i.e., business logic) encoded into the program.

5. Visualization The data provided by the service is finally visualized in the user interface of the client device, e.g., a web browser.

Practically, the tasks are represented by the basic components of Faucet-Sink-Drain Model, the data processing model initially described in Publication V, and the main topic of this thesis. A more detailed description of the components of the model is provided in

Section 4.1, while the tasks carried out by the model are revisited in Section 4.1.2.

System modeling is a way to simplify the design, but even more importantly, it can give a clear and understandable view of the system and its interrelation with others. In the best case, a simple figure can answer the question "How should I implement this system in order to accomplish my requirements?" With the ever growing and more complex systems of systems, understanding the important parts is even more critical.

3.2 PUBLICATIONS IN DETAIL

This section provides an insight into the contribution to this thesis by combining the different topics together. The research done for each of the focus areas is summarized in Table 3.1.

Table 3.1: Summary of research contribution.

	Publications					
Focus Area	I	II	III	IV	V	VI
Acquisition		✓	✓		✓	✓
Storage	✓	✓				✓
Identification & Filtering	✓					✓
Processing		✓		✓		
Visualization	✓	✓	✓	✓	✓	✓

The table rows represent the focus area, and the columns represent the publication selected for this compilation thesis (see also Section 1.4). The check marks on the table indicate that this publication is related to the focus area.

ACQUISITION Data acquisition is one of the first steps performed by a system. Without data, there cannot be information, and without information, there is only a little value to the end user.

In Publication VI, a mobile phone utilizing its sensors is an example of raw data collection. Raw data is often some sort of time series data, e.g., periodically collected data points from a certain sensor or sensors. Acquisition additionally relies on the communication of the data from the source to the processing system. However, everything cannot be automated easily, or it is not financially viable to replace measuring devices with the newest technology. Many devices that are available can be networked and contain automatic error detection or monitoring software. However, this is not true for all devices, especially when taking into consideration legacy devices which still have to be

supported. If these devices are seldom used or replacing them would be expensive, alternative approaches are required. (Publication II)

Sometimes further processing is performed directly on the sensing system, but the data still needs to be communicated in a certain way for the receiving end to understand it. Often the data we are interested in has already been collected by another party, but the data format—i.e., the language our system understands—is not suitable for specific use. Data sanitizing, lexical transformations, and other preprocessing operations are almost mandatory on the way to processing data into more useful forms. However, the accountability and integrity of the data cannot be verified if the system operates in a "black box" mode. Therefore, to increase the transparency of a system, it should be able to describe each operation that modified the data.

Furthermore, the topic of data acquisition has been considered in terms of reliable condition measurement as well as the well-being of the people using the system or living in an area where the system is being operated (Publication III). The objective was to generate a reliable portable system that presents measurement data clearly for the monitoring of changes in conditions. In practice, property automation installed in buildings demands continuous monitoring of conditions to work properly. It is indeed crucial to ensure that a property automation system works continuously and flawlessly, or correspondingly. If there are problems in the process, the problem areas can then be identified so that the necessary adjustment and corrective actions can be carried out. The objective of condition monitoring is at a healthy and functional living and/or working environment as well as energy cost control. Additionally, the right conditions, such as indoor air quality and the right temperature, are fundamental as far as comfort and well-being are concerned. (Publication III) Data acquisition was briefly discussed in Publication V regarding the proposed use cases for real-time data gathering and processing.

STORAGE The storage phase can be a straightforward task if there is just a small amount of data to be stored—simply save everything! Realistically speaking this is not a viable approach. Problems may also arise if the original data set is too large so that some data preprocessing has to be done to conserve space. Thus, strategies must be formulated concerning which data and which intermediate results must be preserved, and what can be discarded. Even the discarded data could have been useful in some form at a later point of time, so there should be a mechanism for providing a means of activating the storage of this data. It should also be considered whether attaching the complete data processing cycle (i.e., what has been done to achieve this result) as additional metadata provides a clear advantage. It could potentially reduce the requirement of storing everything by offering

reproducibility, but at the cost of processing everything again as well as increasing the ratio of metadata to actual data.

The storage scenario may depend on the nature of the ingested data and produced results. Is the data produced in real time and required immediately, or can it be processed in batch runs? The answer can help to place the data in the correct storage tier (e.g., in-memory, solid-state, hard disk, tape drive). Usually, the economic preconditions differ from what might be the best solution for the current problem at hand. For example, two of the previous prototypes (Publication II; Publication VI) differ greatly in the amount of data points: the first produces just a few per day, while the latter creates one data point once per second when in use. The utilization of the data, however, is quite low: the first case is checked perhaps a few times per week on a regular basis, keeping the total system load rather low. In the latter case, the data is processed at very rare intervals, but once the data is requested almost all the data needs to be processed. This can create a lot of traffic and other related processing requests, but on average the system might indicate low overall utilization. Knowing what kind of information might be requested beforehand can help to decide if there is a way to process and store preliminary results from the original data, and serve it instead of the whole data set.

IDENTIFICATION AND FILTERING For efficient processing of the data, it has to be identified and indexed properly; usually it is also tagged with metadata. It is also useful if the selected data contains only the required data as it will reduce, for example, the required bandwidth and memory footprint. However, making the data too clean can be just as problematic if additional queries have to be performed to find missing information. For example, it can be hard to deduce what kind of shaking is due to road conditions and what kind is due to problems in the car itself (Publication VI).

In modern business, one of the key issues is ubiquitous access to the organization's back-end system information, company data, and process infrastructure. Enabling access to corporate information anywhere and anytime is essential. We can state that, in principle, advanced mobile technology facilitates the real-time collecting and sharing of data and its storage directly in the company's information systems, so that the updated data becomes available to everyone. (Publication I)

PROCESSING The processing step can be difficult to define and position exactly. For example, even the data that is ready to be presented still requires some post-processing on the device conducting visualization. This is the case in the prototypes in Publication II and Publication IV where both end user web applications request the data, but perform a few more steps before visualization. In the first case,

a daily 24-hour rate of change as well as the price of energy are calculated, and in the latter, the data is inspected for the determination of hazardous concentration of heavy metal substances. For the latter case, the processing is basically done in the correct phase (according to the typical model presented in Figure 3.2) but in the first case, some extra processing occurs again in the visualization step.

In theory, the data that is prepared should not need any more processing, but in practice some operations are still done on the visualization end. This means that each and every device unnecessarily duplicates the processing work. In this particular case (Publication II), the incurred processing cost is almost insignificant. However, if the processing task is more demanding, then the accumulated amount of time and energy may become significant. These are the hidden costs which can be challenging (even impossible) to calculate in budgeting and resourcing. For visualization purposes, the balance between repeated processing and customization should be supported by the data processing back-end. Perhaps there will be techniques in the future that allow direct visualization with a minimal amount of extra processing, but that is beyond the scope of this thesis.

VISUALIZATION Visualization of data meaningfully is becoming more and more important as the amount of data increases. Good visualization also helps the user of the data to interpret it more easily. With visualization, more useful knowledge can be extracted from data. Visualizations of data allow the user to gain insight into the data and come up with new hypotheses and research questions about the phenomena behind the data. (Publication IV)

If required, a system can provide alternative visualization methods, such as those practiced in Publication II. The system allows the user to examine the collected data quickly on the mobile client application (simple view) and more thoroughly using the management user interface (full chart including analysis information) once the user has returned to their office. The rationale for limiting client application features was to keep it as simple as possible and therefore reduce the maintenance required for the application. It also helps to keep the device small enough to carry around and to use for entering data. Also, the employee typing in the data might be more interested in seeing if the figures show any unexpected highs or lows, so he/she can react to the situation more quickly.

In another example (Publication IV), the prototype application for reviewing the quality of water was implemented as a HyperText Markup Language 5 (HTML5) web page utilizing JavaScript for interactive features. The goal of the prototype was to provide an easy-to-access and a simple user interface with a minimal amount of configurable options. A simple classification (safe or toxic) for different categories should be clear indication for the user to determine water quality. The

application can be seen as an improvement over spreadsheets as it can speed up the analysis of the data by skipping the export and the import phase completely. (Publication IV)

The system introduced in Publication III assists maintenance staff and also supports managers who are responsible for ensuring the correct operation of the devices. Thus, the system is one step toward more reliable measurement data, and it also improves the visual presentation of collected condition data for analysis. The ability of maintenance staff to monitor changes in real time has also been investigated in the public swimming pool data gathering case (Publication II): "The system developed facilitates the maintenance staff's work in registering and recording the measurement information as well as real-time tracking of usage information and perception of possible anomalous consumption situations."

An attempt at adaptive software UIs was made with the system described in Publication I. In this case, a multiplatform framework was created and introduced, which provided a means for a modifiable UI utilizing adapter software. The goal of the framework was to allow users to create applications, distribute applications to desired clients, use applications on any device with compatible adapter software, and gather data. In Publication I it was anticipated that adapter software "could be one of the key emerging technologies in the information technology sector in relation to mobile devices and telecommunications" and that it had "great potential for enhancing information transfer in mobile devices, and consequently for improving information management." Soon afterwards, a standard for HTML5 emerged and paved the way for sophisticated web applications. Since then the web applications have become very common, but the claim about adapter software was not realized. However, in a broader sense, the modern web browsers can be seen to resemble the adapter software approach described in Publication I.

4

THE DATA PROCESSING MODEL

This chapter presents a detailed data processing model for controllable and systematic data source management. Briefly, the model contains the idea of how the data is processed in software systems and aims at the logical organization of various data streams. The model is iteratively developed and experimented in prototypes following the DSRM process. The model, also known as Faucet-Sink-Drain Model, describes the foundational building blocks and relationships. The MDS component framework is the resulting software artifact that was designed and realized during the prototyping process. The framework itself demonstrates the initial capabilities of Faucet-Sink-Drain Model as generalizable software for systematic control of the data flows in a software application.

4.1 FAUCET-SINK-DRAIN MODEL

This section focuses on introducing the data processing model called Faucet-Sink-Drain Model. First, an overview and motivational look at the model is given. After this, the individual components of the model are examined in more detail. Lastly, the reference architecture of the MDS component framework is described.

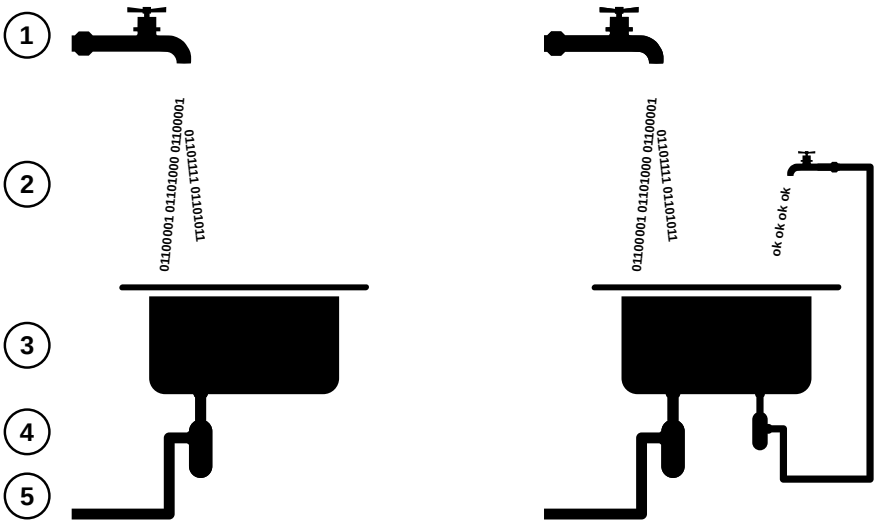
4.1.1 *Overview*

The idea behind the Faucet-Sink-Drain Model was envisioned by the author while washing dishes in a kitchen sink. The water enters the system from a predetermined point by the user who controls the volume and heat of the water stream. Then the water is combined and mixed with the previous contents of the sink. Together with a "catalyst" (such as a detergent), a tool (e.g., a sponge), and external force, the water can clean the contents of the sink by separating or dissolving the unwanted substances. Finally, the fully "processed" water is released from the system from a predetermined point of exit, into the drainage system, by opening the filter cap in the sink. For additional effectiveness, each step could also be interrupted, observed, and inspected by the user.

DESIGN AND RATIONALE OF THE MODEL Firstly, as briefly indicated above, data processing can be modeled with a water piping apparatus consisting of five components: faucets, streams, sink, sieves, and drains. The data flows 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 generates new streams to be stored in the sink. (Publication V; Publication VI)

Secondly, as stated in Publication VI, it is possible to create highly practical and interoperable applications with the use of fundamentally simple models. This can result in the improvement of the overall quality of the software. In Publication VI, a combination of models for 1) data gathering and 2) analysis of the gathered data was studied to enable effective data processing of large data sets. When these points are fused, it should be possible to create data processing applications with a "natural-like" work flow that assists in achieving the necessary information for interpretation of gathered data and decision-making.

The basic overview in Figure 4.1a indicates the mandatory components of the model's processing cycle. Working from top to bottom, the five key components of the model are as follows (prefix `Data` is omitted for clarity): 1) faucet, 2) stream, 3) sink, 4) sieve, and 5) drain.



(a) Simple case. (b) Extended case.

Figure 4.1: Conceptual model of data processing. (Publication V)

The first component is a *faucet*, which appears to be open and running two *streams*. Any stream can flow into a *sink*, and eventually can be captured by any *sieve* (i.e., a filter device) attached to the bottom of the sink. The sieve provides a connection to the last component, a *drain*. (Publication V) The purpose of a faucet is to provide a controllable entry point of data to the system through the connection to a drain. The drain represents the data transfer between system(s) and provides an interface for faucets to attach to. Streams emitted by faucets are

representations of data, and the sink is analogous to a database. The sieve collects and combines the data available in the sink, and emits the requested data forward for further processing.

When the data is combined together and processed with expertise, the result will provide more data, which in turn can be processed again with some other data. If the data, selection criteria, and processing methods are chosen wisely, the data refinement cycle (i.e., *processing cycle*) can eventually provide information that is valuable to its user. (Publication V) A system that is able to perform such data processing on a wide range of source data must therefore be generic, customizable, and extendable. The model can provide all of these features with an extension of the feedback mechanism, as shown in Figure 4.1b.

The model accepts data streams of any kind from any data source in order to provide generic features. The requirement is that the data streams have been supplied with the necessary metadata (i.e., a few basic pieces of information encoded into `DataStream`). This metadata may be given by the user (e.g., when operating with raw data) or be automatically generated by faucets in the subsequent processing cycles. Customization means that the model allows processing of any data with the ability to interpret the selected source data in different formats. For example, interpreting a string "11" as number 11, and for more advanced examples selecting several different types of data having a similar timestamp, and combining those together. The system resembles data lakes in being a repository for unstructured data. Any data which is acquired (by any faucet) is in practice unstructured. If the data cannot be interpreted directly, it is the responsibility of the sink and sieves to translate/modify/combine the data into a more useful structured data format for further processing.

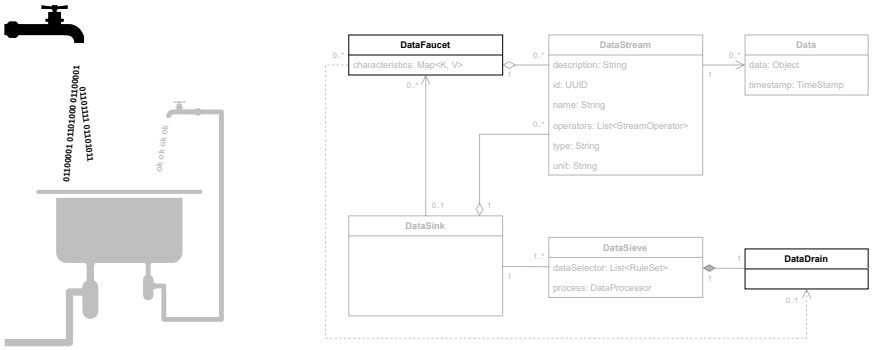
With the feedback mechanism, it is possible to "loop" any of the aforementioned data through any number of processing stages, in theory indefinitely. Lastly, the model does not have to be of a single instance, but there may be multiple data sources (i.e., drainpipes) to attach "our" data faucet onto. This allows, for example, the importing of preprocessed data sets, and utilizing Open Data provided by multiple other users. The same data could also be forwarded to two different instances with slightly different processing algorithms for purposes of result comparison. Lastly, to keep everything under control, the model expects that every data stream—including data and the accompanying metadata—should be accountable in terms of origin and all the operations which have been performed. In theory, the accountability should also improve the reusability of the data.

4.1.2 *Abstraction: Model in Brief*

In the following section, the model is compared to the elements explained in Section 3.1 to establish the responsibilities of each com-

ponent. Each figure in this subsection is composed of subfigures (a) and (b). Subfigure (a) is similar to Figure 4.1b with added transparency to highlight the component. Subfigure (b) is the architectural class diagram of the case with a corresponding highlight. Text in these figures is purposefully left illegible, however, the legible version of the class diagram (i.e., architecture) will be discussed in Section 4.1.3.

The *Acquisition* step in Figure 4.2 represents the entry point of the data. The faucet itself specifies the attached and controlled point where the data streams of the providing source are allowed to flow into the system. From the model point of view, the origin of the acquired data (specifically, data streams) can be treated similarly in all cases; the data can be from external sources (the leftmost faucet) or be reprocessed by the system itself (the rightmost faucet). In Figure 4.2a, the leftmost faucet is highlighted to indicate that the external source is active. The most important feature is the type structure in which the data streams (which are the enclosing constructions of the actual data) store the information of where the data comes from and what kinds of operations have been done to the data. By inspecting the structure, it is possible to trace the origin of any piece of data inside the model. Classes involved in this step are `DataFaucet` and partially `DataDrain`.

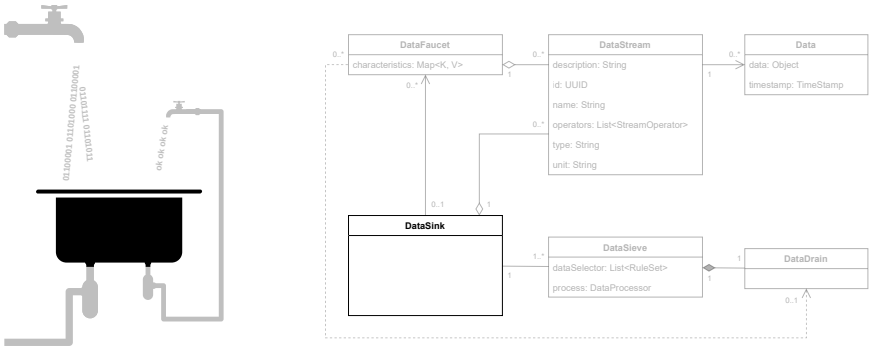


(a) Acquisition step. (b) Reference architecture.

Figure 4.2: Data acquisition.

The *Storage* step is mainly handled by the data sink as shown in Figure 4.3. The model, however, intentionally does not specify how the data is to be physically stored, so each implementation is free to make the most appropriate choice of solution. The implementation can use and support any logical storage retention plan, for example, but not limited to: 1) keep everything in the sink, 2) keep only the data requested by the sieves, 3) have the data consumed immediately by the n-th applicable sieve, or 4) remove the data once there are no dangling dependencies by any sieve. There should be options to specify configurations that override the default storage retention settings. If the sink needs to do any special tasks (for example, accumulate, calculate, or compare), it can be instructed to do so using the `StreamOperators` attached to the data streams. Operators can be injected into the stream

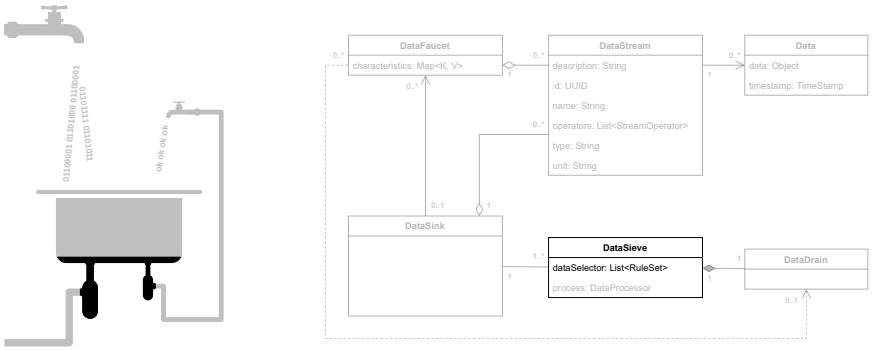
by any other component action, for example in the faucet by user intervention, in the sieve by the query developer, or in the drain by the system. The main class involved in this step is `DataSink`.



(a) Storage step. (b) Reference architecture.

Figure 4.3: Data storage.

The *Identification and filtering* step will be performed on each data sieve that is attached to the sink (see Figure 4.4). Sieves always check any incoming streams and data, and depending on the implementation this could be performed in parallel or sequentially. Race conditions and the order of data entering the sieves might also matter if the sieve system is configured to allow simultaneous data access or not. A sieve has a predefined set of rules and criteria that define what kind of data it expects and requires from the data sink. In this sense, the sieve is in practice a query tool for the data. The main class involved in this step is `DataSieve`, along with `RuleSet`.

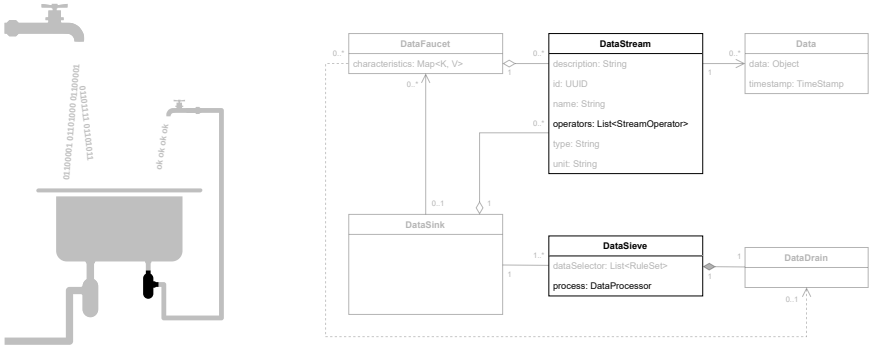


(a) Identification step. (b) Reference architecture.

Figure 4.4: Data identification and filtering.

In the *Processing* step, the bulk of the processing is carried out inside data sieves. However, some parts of the post-processing may occur in other parts of the system due to the use of `StreamOperators`. As illustrated in Figure 4.5, the processing happens only in the rightmost sieve as if it was activated by the previous step to find suitable data for that particular sieve. As explained in the previous step, sieves select and prepare the data for further processing. Sieves are also

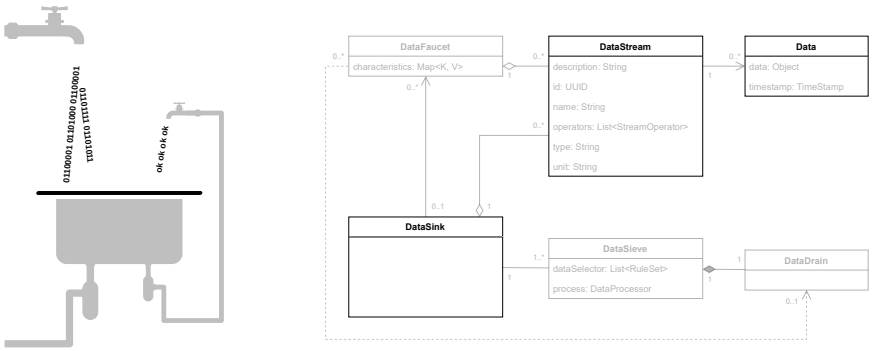
the first components which can inject additional operations into the data streams emerging from them. Once the data specified by the data stream reaches the faucet, the accompanying operations are performed. The generated results are primarily intended for accumulating information usable for reporting purposes. However, it should be possible to feed-forward the results to other data sieves to generate new data and/or data streams. The main classes involved in this step are `DataSieve` together with `DataProcessor` and `StreamOperator`.



(a) Processing step. (b) Reference architecture.

Figure 4.5: Data processing.

The *Visualization* step involves the presentation of the data emerging from the faucets. This requires (read-only) access to the data, and it must be supported by providing the necessary interfaces so that the requesting program/application/user can get a view to the data residing in the sink (see Figure 4.6). The model can provide specialized UIs for visualizing the data, but it is not obligatory. In the previous version of the model (Publication V), the visualizations were explicitly included in the sink, but on further consideration it was made optional. Thus, the realization can choose whether to include it or not. The prototype system introduced in this thesis does not include such visualizations. The main class involved in this step would be `DataSink`, accompanied by `DataStream` and `Data`.



(a) Visualization step. (b) Reference architecture.

Figure 4.6: Data visualization.

In summary, the model can provide a lot of opportunities for data processing. However, it does have the downside of allowing the formulation of meaningless operations. This could include cases such as selecting all numeric & textual data to perform an arithmetic sum operation on the unchecked inputs. Therefore, special care should be taken in the choice of data selectors and operations, for example by blocking the most commonly known errors. Other approaches to mitigate the issue are to perform a smart error checking of operations, or by producing a live preview of the input and output data.

4.1.3 *Implementation: Manageable Data Sources Framework*

This section describes the design choices that were made when the MDS component framework was being realized. The framework aims to help the processing, combination, and management of data sources by making them controllable and systematic. With a systematic approach, the accountability of data (i.e., what steps were taken to achieve the result) should become more evident and reproducible. The model is applied for MDS in order to create a usable system for the handling and processing of data. MDS covers the creation of data, includes the search, selection, and combination of any input data, and finishes with the outcome of the processed data such as a report. Furthermore, as defined by the model, MDS gives the control of the data to the users by letting them choose and manage the desired features such as data sources, processors, and so on. (Publication V)

The reference architecture of the core components of the MDS component framework is depicted in Figure 4.7. This high level view identifies the associations and relationships between core components.

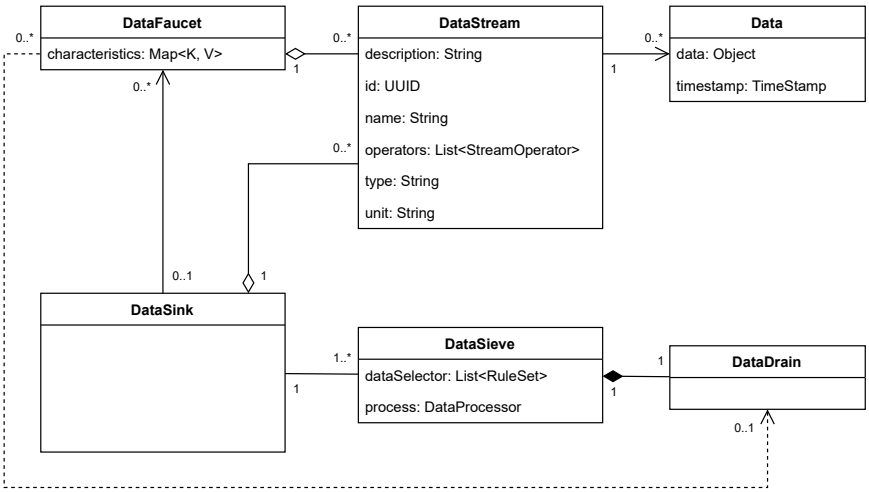


Figure 4.7: Reference architecture. (adapted from Publication V)

One important aspect is that this depiction should not be interpreted as a closed loop system; rather, it is a logical representation of the

interrelations between the components shown. In fact, there could exist as many instances as required, and there could be connections between different instances when deemed necessary. For example, DataFaucet can receive data from DataDrain created and defined by another user. (Publication V) This is possible if the said DataDrain has been published and shared for public use. Additionally, many faucets can be attached to one drain, and these connections can be kept fully unidirectional as the drain does not need any information from the faucets connected to it. It is possible, as demonstrated in Section 4.3, for the drain to have certain interfaces for a data query to facilitate the retrieval of a more or less partial set of data from the related DataSink.

DEFINITIONS AND SPECIFICATIONS The key concepts of the MDS component framework are presented here with the necessary amendments and clarifications (compared to the original listing from Publication V). The following list contains the fields and concepts which are present in Figure 4.7. The exception to this rule is that simple fields (e.g., description, ID, name) and classes (String, Object, TimeStamp, List, Map, UUID) which are considered as being standard plain data types are not included.

DATA The component that represents the smallest particle that builds up a `DataStream`. It is basically of any type and format, and includes the timestamp when the data was created. Each piece of data is associated to exactly one `DataStream` but the association is unidirectional, thus the data is not aware of its owner.

DATASTREAM The component represents information and its presentation. It uniformly describes the type, format, and origin of the `Data`. The data stream has $0 \ldots n$ `Data`, and is associated to exactly one `DataFaucet`.

DATAFAUCET This component is the source of all content, and defines a uniform protocol to access the data. It specifies the characteristic metadata of the source such as location, update interval, previous processing history, and so on. The faucet has $0 \ldots n$ `DataStreams`.

DATASIEVE Each `DataSieve` component has the ability to filter all of the data inside the sink, select the most interesting streams, and process and combine them into new streams. Sieves may also translate the selected data into a more useful structured data formats. The processed output of the sieve is transmitted to the attached drain. The sieve has exactly one `DataDrain`.

DATADRAIN The drain component is a transmission mechanism for the data. If left unconnected, the data is simply discarded. Multiple faucets can be attached to a single drain, and all of these faucets will receive the same data. The drain may have $0 \ldots n$ `DataFaucets`.

DATASINK The central component where the data resides and can be observed. The sink will share the ownership of a `DataStream` once it has been emitted from the `DataFaucet`. The sink may store and accumulate the data, or simply update the current data instance in the sink based on the `StreamOperator` defined on each stream. The `DataSink` knows $0 \dots n$ `DataFaucets`, and has $0 \dots n$ `DataStreams`. There must be at least one $(1 \dots n)$ `DataSieve` to prevent "information overflow" in the sink.

DATAPROCESSOR A set of rules for processing `DataStreams` by the `DataSieve`. It defines the means necessary to create new output data from the selected input data (e.g., addition, subtraction, multiplication, logical AND & OR, set combinations, concatenations, and so on). Variation point for application specific tasks.

RULESET A set of rules for selecting the desired `DataStreams`.

STREAMOPERATOR This controls how the `Data` of a `DataStream` has to be handled once a new data point emerges. Specifies how to formulate a result, and how it should be stored. Results can be accumulated into a single updating variable, or concatenated to form a time series data. `StreamOperator` can have multiple sub-operations carrying out tasks such as differential and integral calculus, percentage calculations, continuous average, and so on.

TYPE A string representing what kind of `DataStream` is in question.

UNIT A fine-grained string which can be used to identify `DataStream` elements. For example, it can be used to indicate measurement (e.g., SI unit), calculations, and so on. For more specialized classifications, it is possible to use any (purposeful and imaginable) scheme such as Multipurpose Internet Mail Extensions (MIME).

Table 4.1 summarizes the major terminological changes between the framework presented previously in Publication V and this thesis. These changes are made for more accurate representation. Visualizations (visualizers) are no longer explicitly included as it can be better to give the responsibility of those features to another component or system.

Table 4.1: Summary of terminological changes.

Changed Concept	Now	Previously
DataFaucet::characteristics	characteristics	physicalInformation
DataStream::type	String	Type
DataSieve::dataSelectors	RuleSet	Criteria
Visualizations	N/A	Visualizer

Data formats for information sharing and various settings files were also considered in Publication V, but the specific implementation details will remain out of the scope of this version of the system.

4.2 USE CASES

The core features of the Faucet-Sink-Drain Model can be demonstrated with the help of three use cases: 1) reading and processing a data file, 2) combination with external data, and 3) utilizing multiple data sources. The complexity of the described use cases increases gradually, beginning from the most simple case.

The base data for the experiments was previously collected during another research project, and utilized in Publication VI. This data set will be referred to as *shock data* from now on. Shock data is basically road condition data collected by leveraging crowdsourcing and accelerometer sensors on mobile phones. The data set was quite ideal for proof-of-concept testing as it is almost completely numerical, thus easy to process, has a small number of different attributes, and a sufficiently big amount of data points (approximately 483 thousand rows in total, and 146 thousand rows after sanitization and deduplication). The results provided by the application could also be verified by referencing to the calculations made by other means (mainly, by various manually crafted spreadsheet functions).

The experimentation of the third use case is based on a system which is used to detect water leaks in a large building complex. The schematics and the technical implementation for detecting water leaks is described in more detail in paper by Rantanen *et al.* (2021). In summary, that system consists of a few interesting raw data sources; the actual water flow status detected by measuring the temperature fluctuations of the water pipe; the usage statistics of the building; and the blueprints or details of the layout of the building. Ultimately, the system would produce a simple status view for the user. The status indicator was chosen to be a traffic light indicator where green color means all good condition, yellow color indicates a minor problem in the system, and red color is an indication of a major problem, such as a detected water leakage.

USE CASE 1: READING AND PROCESSING A DATA FILE In this use case, the system is given a task based on the shock data set file. In the scenario a researcher works with their half yearly released research data, which has been exported from a database into a Comma-separated Values (CSV) file. Now the researcher needs some key figures to be extracted from the data but the previously used spreadsheet with the associated formulas has once again gone missing. Additionally, due to the fact that it is a relatively small task and infrequently needed, no real effort to automate the reporting system has been carried out. It can also be that automation has not been done because the perceived work load of developing such system is too high. Therefore, the challenge is to overcome the opportunity costs of developing an automated system compared with manually conducted (repetitive) work.

The goal of this use case is quite simple: read the contents of a file and apply the necessary processing steps to produce and retrieve statistical information from the data. Basically, the procedure to achieve the goal can be separated into five steps: 1) read a CSV formatted file on the local file system, 2) read and parse the contents of the file, 3) select the interesting pieces of data, 4) process the selected data with appropriate functions and methods, and 5) print out the results. These steps essentially represent the tasks portrayed previously (e.g., in Figure 3.2 and Figures 4.2–4.6). The procedure of how this was implemented on the prototype system is described in Section 4.3.2.

The implication of this use case is that any Faucet-Sink-Drain Model system is able to ingest and process various kinds of data. This is possible due to model guides to work in a specific way as systems are specialized to fulfill the processing tasks. In the context of the prototype system, it would be necessary to implement components such as file parsers, column selection rules, mathematical algorithms for data processors, and so on. The framework is designed in such a way that many of the components can be instantiated again in different scenarios for the increased reusability (e.g., `CSVParser` and `FileFaucet`). In theory, the framework can operate in an incremental processing mode as it knows which data is new and unprocessed. Although it might be imperceptible in this use case, this mode may provide more savings of computational resources in different contexts.

Generic data processing could be adopted by many different software applications, and for example, make data sharing between applications more streamlined. As a result, the emphasis of data processing development efforts could shift more toward the creation of useful content and information instead of the handling of individual data points. This use case has its roots in Publication VI, where the model was applied and documented for the first time.

USE CASE 2: COMBINATION WITH EXTERNAL DATA In this "drawing board exercise," the aforementioned researcher needs to append weather data to the shock data set for discovering the road condition of any given data point. The shock data set already contains the exact geographical location and timestamp, thus the remaining challenge is how the data processing software can be instructed to collect and combine the weather data efficiently enough.

The goal of this use case is to extend the previous case by combining additional external data to the locally available data. This represents a typical situation where some resources are provided by external services. By extension, it enables Faucet-Sink-Drain Model systems to connect with different data sources. Generally, the traditional ways of solving the problem include obtaining the weather data from a third party service by adapting our own system to their data formats

and APIs. The adaptation usually requires additional source code development and adds to the general complexity of the application.

Faucet-Sink-Drain Model provide benefit to a system by reducing the complexity data handling during the data import phase: by treating all data similarly. From the model point of view a data faucet is an abstraction of any (internal or external) data source. The faucet defines a uniform interface to access the data. The faucet abstraction enables the user to use similar source code on multiple, varying data sources. Therefore, any emerging data is readily available for utilization.

In this case, the researcher needs to define a new `DataFaucet`, which points to the requested `DataStream` from the `DataDrain` of a weather service. This means that the software retrieves the corresponding weather data from the server through the supplied data query interfaces, handled behind the scenes by the faucet abstraction. As a result, the emerging data stream will contain the appropriate (time- and location-based) weather information for further utilization. For example, the processed shock data and weather information may be bundled together, and served thru another `DataDrain` service. For traceability and verification purposes, the drain should provide a list of operations that were utilized to reach the result.

The second use case is a small increment to the idea in first use case, but it provides plenty of extended functionality for the system. These will be very useful on reusing the shock data—or any other data source—in different scenarios and systems. However, it was not fully implemented due to constraints on working resources. In bridging a gap between a Faucet-Sink-Drain Model and third party systems, it should be technically possible to implement a wrapper service that creates an artificial `DataDrain` interface between the said systems.

USE CASE 3: UTILIZING MULTIPLE DATA SOURCES In the third use case, the desired data varies significantly from each other. Their data format, method of data delivery, among others, need to be dealt with before getting started to operate with the data. Thus each source would need to have a custom-made data import implementation. After the source data is made accessible, data processing algorithms are able to work. It is crucial to have a common data access format (i.e., an abstraction layer) to make the data import task less complicated.

The goal of the use case is to demonstrate the extendable multipurpose nature of the Faucet-Sink-Drain Model, and its ability to work with multiple generic data sources. This can be achieved while following the principles of the model, and also means that interoperability between different Faucet-Sink-Drain Model systems can be seamless.

An example system expressed in the context of Faucet-Sink-Drain Model is depicted in Figure 4.8. The figure has three main parts of data flows, where the last data flow is being utilized by an external system: 1) raw data combine processing part which includes the group of three

faucets on the left, 2) generic component processing part consisting of one faucet in the middle, and finally 3) information processing part which is the last faucet on the right. In fact, there could be multiple faucets in the last part, depending on the complexity of the required operation. The data stream containing the processed information flows to the sink similarly like all other data.

The combine processing part includes the following data streams: a) sensor location data and floor plan map, b) building's locked/unlocked status which conveys the estimated working hours for setting normal and abnormal changes (e.g., differential over time; $\frac{dx}{dt}$), and c) time series data from the temperature sensors.

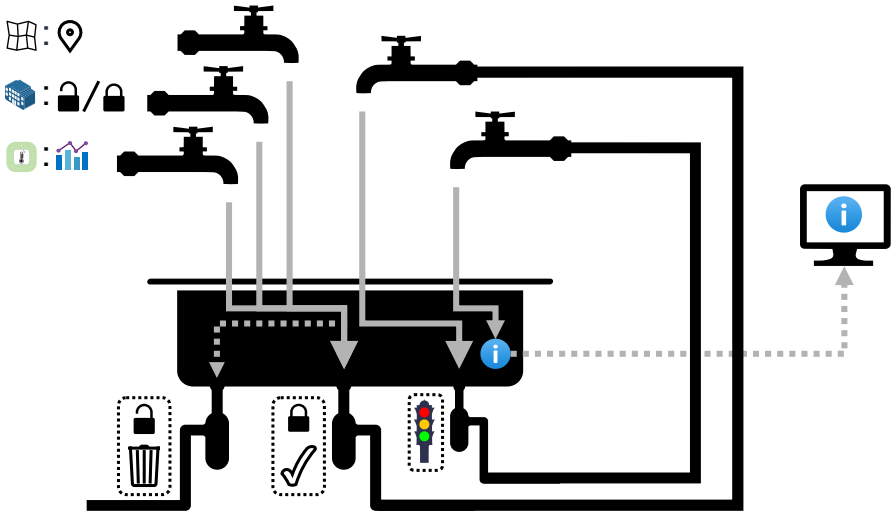


Figure 4.8: Model of data flows.

The combine process continues as follows. The data will be processed by the combinatory functions in the second sieve in the middle. The initial selection is based on the building's locked status (i.e., active/inactive usage). At certain hours, the building is in a stage of low demand use, and that is the time when abnormal leaks can be detected. All other raw data, which are not captured nor needed, are discarded thru the main drain on the left, as indicated by the dashed line and the unlocked state. The sieve will then transfer the combined data stream to the faucet in the middle. The data stream contains sufficient amount of information for the next step, traffic sign processing.

The output of the middle faucet is a stream which includes all three data sources as a one logical set. In this second part, or traffic sign processing part, the rightmost sieve is programmed to collect the previously combined data stream. It processes the data with series of functions in order to generate data for a "traffic light" indicator. The traffic sign process shown in Figure 4.8 could actually utilize multiple sieve-drain-faucet cycles instead of just the one that is depicted.

The third part, information processing, depicts the finalized product of the data processing cycle. In this step, the information stream

(encircled by letter "i") is the output stream of the rightmost faucet. At this point, the outputted data stream has been chosen for monitoring by an external view port.

4.3 PROTOTYPE SYSTEM

The purpose of the prototype system introduced in this section is to gain an empirical basis of the model in terms of applicability and feasibility in anticipated use case scenarios. It also serves as a starting point for determining the level of reusability and generalization of the implementation.

4.3.1 *Overview*

The basic structure of the prototype follows the architecture which was described in Figure 4.7. Specifically, the prototype system is a representation of use case 1: reading and processing a data file. The data flow diagram of the system is shown in Figure 4.9.

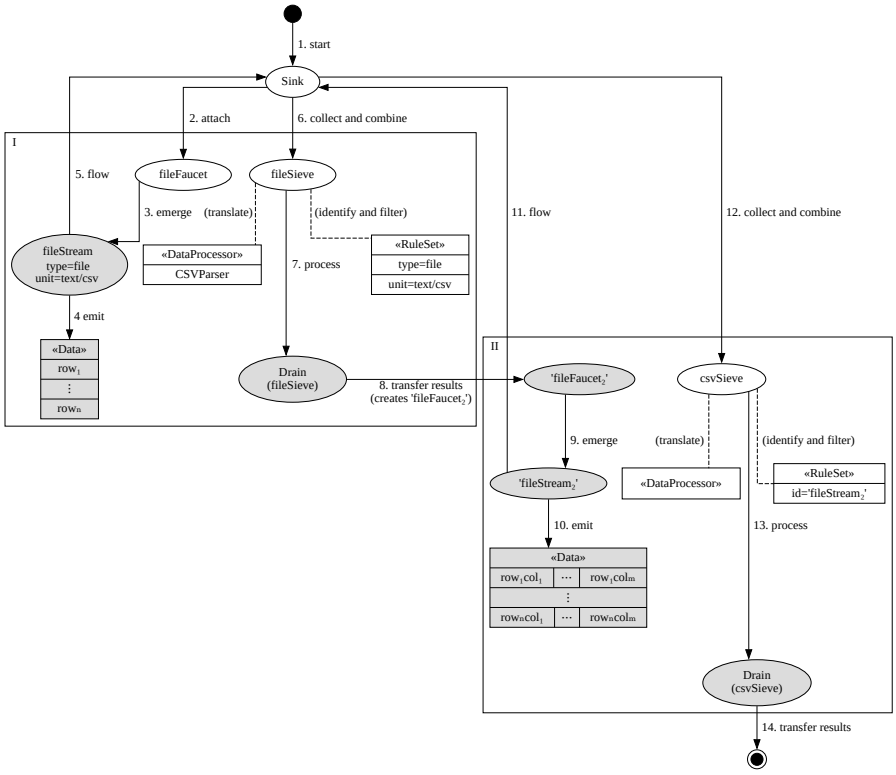


Figure 4.9: Overview of prototype data flows.

Each step is consecutively numbered (i.e., `1. start`) and is accompanied by a verb which describes what is happening in that step. The verbs use the semantics of the actions described in Section 4.1. Shapes in the figure are denoted as follows: 1) ellipse is an object which is one of the five key components, 2) rectangles are objects belonging to

a certain component, and 3) large rectangles around shapes designate a clustered package. The shapes colored in light gray are like their normal equivalent, but can be implicitly instantiated by the prototype system (e.g., the parent object contains all information needed to create a complete object of such type). Text written in parentheses gives an extra remark for the corresponding section.

The starting point (1) of this diagram is just before attaching the first faucet to the sink. The next steps (2–5) are related to the internal handling of data on the first faucet which provides "raw" data to the system. The faucet in question is specialized in reading files from the local file system. When the said *fileFaucet* is attached, a single *fileStream* emerges from it, and the relevant file data will be emitted. In this case, stream is populated with metadata such as `type` and `unit`, and each file row is treated as a separate piece of data. The stream and data flow into the sink.

Once the data flows into the sink, the next steps (6–7) are handled by a data sieve. The data stream is identified by a *fileSieve* due to its `RuleSet` being suitable enough (actually, it is identical with the *fileStream*). Then it is time for processing the captured stream or streams. The *fileSieve* has a `DataProcessor` which is of type `CSVParser` that can translate raw data rows into a processable format according to CSV rules. The processed stream and data is outputted to a drain for further usage (8). The newly created *'fileFaucet*$_2$*'* can be seen in the next cluster but firstly, let us examine the first cluster (`I`) more closely. All shapes in it forms the archetype of a processing cycle, and represent the basic structure that every other key component will follow. The application specific variation is possible as can be seen on the subsequent cluster (`II`).

The faucet created in step 8 is automatically created according to the information provided by the drain. The same attach, emerge, emit and flow steps are represented by steps 9–11. For illustration purposes, to indicate it is the second cycle, the names have been appended with a subscript number two. The most obvious difference with this faucet is in the data. The data is in structured format now, containing the column name information which was present in the CSV file. It was translated by the `CSVParser` into a format that is native to the prototype system, thus it is now readily processable by any subsequent step. In fact, `CSVParser` is an extension of internally utilized `DefaultProcessor` which is a minimum implementation to define the required functionalities for `DataProcessors`. The same occurs with other classes with a prefix `Default` in their name.

The logic may seem quite complicated, but there is a reason to do so. Firstly, it creates a well-formed structure and interfaces for carrying out (simple or advanced) work with source data. Then, the structure allows reusable components, and lastly generism in data processing enables application specific variation on the same source data. This

is exercised in the last remaining steps (12–14) by *csvSieve* where the second file stream is captured, processed, and transferred forward. The end state concludes the use case 1 and can be used as the start for the use case 2 (see Section 4.2).

4.3.2 *Detailed Description*

In this section, the prototype system is detailed from three views. Firstly, how it is initialized, then in which way it can operate with data, and lastly how it reports results.

Listing 4.1 and Listing 4.2 provide a source code (written in TypeScript) level overview of how an application could utilize the MDS component framework from a development point of view. The listed source code is commented inline, but for added clarity, the code is examined in more detail below. At this point it might be worthwhile recalling the high level tasks portrayed previously in Section 3.1.2, Section 3.2, and in Section 4.1.2 (see Figures 4.2–4.6). The binding of these tasks in the implementation is also explained below. The former list concentrates on preparing the system for receiving data while the latter indicates how a certain data stream will be processed.

INITIALIZING THE SYSTEM In Listing 4.1, the system is being prepared for use by instantiating the components appearing in Faucet-Sink-Drain Model. Many components instantiate the necessary objects by themselves behind the scenes. For example, the developer can leave the creation of `DataDrain` to the framework most of the time. Another example with data processors is the creation of a new logical `DataStream` and generation of new `Data`. These are implicitly redirected to the *sink* instance through the new instances of `DataDrain` and `DataFaucet`. The typical code for application initialization is also omitted from the source code listing. The source code in Listing 4.1 consists of three logical sections distributed followingly:

1. Prepare `DataSink` & `DataFaucet` (rows 1–2 and 28)
2. Create different types of `DataSieves` (rows 4–13)
3. Define application specific data selectors (rows 15–26)

The first point fulfills two of the tasks, the acquisition and storage steps, while the identification & filtering task is fulfilled by the second and third points (see definitions in Section 4.1.2). On rows 1–2, a *fileSieve* is instantiated with an instruction to look for a file on a local disk from a specified path, followed by getting a handle to the data sink. The data processor should be given to the `DataSieve` so it can operate properly with the received data. Data selection algorithms may be given different criteria depending on the use case. A *fileSieve*, defined on rows 4–9, is instructed to look for data streams declaring they are of type "file" and having a content type "text/csv" (rows

15–21). In this case, the selection will be targeted to the output of a `DataFaucet` *fileFaucet* defined on row 1. Another sieve is instantiated (rows 10–13), which is instructed to look for data streams having a specific identifier. The stream identifier is created automatically by a series of (behind the scenes) drain-faucet functions, initiated by the *fileSieve* (rows 22–26).

Listing 4.1: Initialization of the system.

```
1  const fileFaucet = new FileFaucet("./data.csv");/*create faucet*/
2  const sink = DefaultSink.getInstance(); /* get sink instance */
3
4  /* create a new Data Sieve with pointer to the sink. This sieve
5   * will be specialized in finding file streams. Second parameter
6   * is Data Processor which deserializes contents into (internally
7   * used) JSON data objects. Here it is a specialized CSVParser,
8   * that is able to separate each column into name-value pairs. */
9  const fileSieve = new DefaultSieve(sink, new CSVParser());
10 /* create another Data Sieve with pointer to the sink. This sieve
11  * will be specialized in finding data streams by specified
12  * identifiers. Creates a basic Data Processor automatically. */
13 const csvSieve = new DefaultSieve(sink, null);
14
15 /* prepare File Sieve to find streams with combinatory rule:
16  * - type of the stream is "file" (defined type of File Faucet)
17  * - unit of the stream is "text/csv" (detected by File Faucet)*/
18 const fileSelector = new RuleSet(); //each rule in set must match
19 fileSieve.addNewCriteria("type", "file", fileSelector);
20 fileSieve.addNewCriteria("unit", "text/csv", fileSelector);
21 sink.addSieve(fileSieve); /* explicitly attach File Sieve */
22 /* prepare CSV Sieve with just a single rule:
23  * - select a specific stream which will be created by the
24  *   fileSieve. Creates an empty Rule Set automatically. */
25 csvSieve.addNewCriteria("id", fileSieve.faucet().stream(), null);
26 sink.addSieve(csvSieve); /* explicitly attach CSV Sieve */
27
28 sink.attachFaucet(fileFaucet); /* finally attach File Faucet */
```

Theoretically, a sieve can select the matching data from any available source by any selection criteria. It enables a flexible way to choose (previously processed) `DataStreams`, improving reusability. For example, reports created from the source data often diverge only after several identical stages have been conducted. Therefore, different operations can benefit (e.g., conserve computational resources) from utilizing a common ancestor stream together.

At this point, the traceable operations may become useful, as traceability can further support the decision on selecting useful types of data streams. For example, if we know the data has been thru a well-known and tested set of operations, the resulting data is more likely to be consistent and error-free. In the prototype system, the accountability of data is implemented by the means of a deterministic hash function.

It is derived from a deliberately chosen set of data stream information variables from any previous and current processing cycles. The result of the hash function is a signature, calculated and distributed forward by the data drain. The signature can be used as a selection criteria in the sieve. As hash functions are one-way only, the criteria needs to be supplied a pre-calculated signature. The pre-calculation is possible because all affecting variables are known in advance, or in case of external data, perhaps the original source already provided their own signature. A divergence from the signature means that data had something which was not expected. Simply put, the current implementation was good enough for conducting proof-of-concept testing, but a more complete method to verify the accountability of data needs to be implemented.

Before row 28 has been executed, the system (i.e., the data sink) will have been in an apparently idle state—just waiting for any data to appear in the sink. The status changes as soon as *fileFaucet* is attached to the sink. This represents the figurative action of the valve of the faucet being opened. Once the "valve" is opened, the data will be allowed to flow into the system for further processing. Each time new data emerges from any faucet into the sink (initiated either by a server push or by a client pull), the system will begin the selection and processing cycle as instructed.

OPERATING WITH THE DATA The source code in Listing 4.2 shows an example of defining a processing step for any `DataStream` object. In terms of the tasks explained in Section 4.1.2, this fulfills the processing task. The further breakdown of the example shows that it consists of three distinct sections:

1. Instantiate `StreamOperator` (rows 1 and 2)
2. Add different logical operations and related data column(s). The example uses just one data column for reasons of clarity (rows 4–12)
3. Define the desired processing functions and fine-grained requirements for each operation (rows 14–27)

The source code shown here focuses on a specific instance of `DataStream` which is ought to be captured by the *csvSieve* (defined on rows 10–13 and 22–26 in Listing 4.1). The purpose is to collect and maintain the relevant processing information for the next step, in this case for reporting results. Any additional instructions (i.e., `StreamOperator`) could be injected into the `DataStream` at practically any point between `DataSieve` and `DataSink`. They will be executed at `DataSink` whenever the sink receives new data belonging to this data stream. The `StreamOperator` class can include more fine-grained filters and terms for combining additional data columns for more advanced processing requirements.

Listing 4.2: Example use of Stream Operators.

```
1  /* prepare Stream Operator with a literal key */
2  const op = new DefaultStreamOperator("Breakdown of Data Points");
3
4  /* prepare 5 simple filters for selecting data for various
5   * operations. The first argument is a literal text identifier.
6   * The following argument(s) specify the needed column(s) and any
7   * additional arguments needed by the corresponding operator. */
8  op.addOperation("Level 0", "computed_level");
9  op.addOperation("Level 1", "computed_level");
10 op.addOperation("Level > 1", "computed_level");
11 op.addOperation("Total Count", "computed_level");
12 op.addOperation("Total Count with Level", "computed_level");
13
14 /* prepare operators for previously filtered data. Operators (2nd
15  * argument) are class-wrapped functions to manipulate data. */
16 op.addOperator("Level 0", Percentage,  // Percentage of rows == 0
17     new Requirement<number>("computed_level", Comparer.IS, 0));
18 op.addOperator("Level 1", Percentage,  // Percentage of rows == 1
19     new Requirement<number>("computed_level", Comparer.IS, 1));
20 op.addOperator("Level > 1", Percentage, // Percentage of rows > 1
21     new Requirement<number>("computed_level",
22         Comparer.GREATER_THAN, 1));
23 op.addOperator("Total Count", CountOf); // Counts amount of rows
24 op.addOperator("Total Count with Level", CountOf,
25     // Counts amount of rows where row (column value exists) >= 0
26     new Requirement<number>("computed_level",
27         Comparer.GREATER_THAN_OR_EQUAL_TO, 0));
```

The first argument in the *addOperation* function (for example, row 8) should be considered as a handle for the corresponding operation. It is used as a key value in the underlying data structure to store the results (see Listing 4.3) as well as operations later on (for example, rows 16 and 17). With TypeScript programming language, the transcompiler can enforce build-time type safety with the use of diamond operators. It can make the debugging of certain errors easier if the data is misinterpreted as strings instead of numbers, and so on.

GENERATING RESULTS A simplified example of the results generated by the application—a very basic form of the visualization step mentioned in Section 4.1.2—can be seen in Listing 4.3. The output in the example is structured as a simple JavaScript Object Notation (JSON) both for illustrative purposes and to be slightly more pleasant for human readers. For machine-to-machine communications, the output should be reworked for better structure and usability. However, the selection of the best (or the most suitable) serialization method will be at the discretion of the user (i.e., developer).

In the JSON output file, the object literal "title" represents the textual description given to the `DefaultStreamOperator`. Each literal key in

object *operation_results* represents the result of the corresponding operation. In the case of the result record of "Level > 1," the percentage of "computed levels" being greater than one is 0.971, and the number of rows is 4691. In the case of `CountOf` records there is only one number included in the result, representing the n of cases found.

Listing 4.3: Example of result output.

```
{
  "title": "Breakdown of Shock Data Points",
  "column": "computed_level",
  "operation_results": {
    "Level 0": { "Percentage": 20.371, "n": 98367 },
    "Level 1": { "Percentage": 9.336, "n": 45083 },
    "Level > 1": { "Percentage": 0.971, "n": 4691 },
    "Total Count": { "CountOf": 482871 },
    "Total Count with Level": { "CountOf": 148141 }
  }
}
```

The numbers shown in the example output (Listing 4.3) do not fully add up to 100 percent. This is due to the omission of data points missing a computed level and also because of the presented figures have been rounded up. The numbers match those available in Publication VI (Table I, column $v \geq 0\ m/s$) because exactly the same source data was utilized here.

4.3.3 *Reusing Technical Constructs*

The prototype system is designed to be extendable by reusing its components. Therefore, it is possible to create new applications with Faucet-Sink-Drain Model based systems, but it can also be utilized in the remodeling of existing systems. An example of such procedure is described in this section while the more detailed description of the system in question is presented in Section 4.2. For the integration with an existing system, a few factors must be taken into consideration. This is required to determine whether the necessary features required by the Faucet-Sink-Drain Model are in place. Those features need to be inspected, and then the most suitable components would be selected to fulfill each of the main level tasks defined by the model.

One needs to identify the relevant system components that perform in the following five tasks as defined in Section 3.1.2 and Section 4.1.2: 1) acquisition, 2) storage, 3) identification and filtering, 4) processing, and 5) visualization. The results of the identification process, and the corresponding components are shown in Table 4.2. The next step is to assign an appropriate task for those components (i.e., the rows in Table 4.2). It is desirable to find as suitable components as possible for all five tasks (i.e., the columns) in order to ensure the compatibility with the Faucet-Sink-Drain Model. The complete mappability is

denoted by • while partial mappability is represented by ○. The task allocation is allowed to overlap between different components, and a component may perform multiple tasks.

Table 4.2: Components of an example system.

Component	*Acquisition*	*Storage*	*Identification & Filtering*	*Processing*	*Visualization*
Data sources (`Faucets`)					
– Building usage statistics	•				
– Building blueprints	•				
– Water flow sensor	•				
Cloud service					
– Database (`Sinks`)		•			
– Algorithms (`Sieves`)			•	•	
– API (`Drains`)	○		○		○
Interfaces					
– Web browser					•

• — mappable ○ — partially mappable

In this case, there have been identified different components that belong to three separate groups: data sources, cloud service, and interfaces. The groups are for clarity reasons, and the name of the group could be just about anything. The components in the data sources group cover the first task, acquisition, while the cloud service spans over the tasks storage, identification & filtering, and processing. The "Service API" component is allocated to three tasks because it has functionalities to support those tasks implicitly. The component "Service Algorithms," however, could be splitted for the benefit of a more fine-grained breakdown of task allocation and more obvious responsibilities. The last group, interfaces, consists of a web browser which is fully mappable for the visualization task.

Finally, the system can be expressed by using the language of Faucet-Sink-Drain Model. The structure of a complete example system is shown in Figure 4.10. The beginning of the figure is on the left side where the aforementioned data sources appear in the `Physical world` environment. Each raw data source is encapsulated in a rectangle in order to distinguish them as independent systems. The encapsulation also denotes the task of acquisition. The wide arrows, or `drains`, from these sources are connected to the three `faucets` on the example

system. It is here where the data enters the Faucet-Sink-Drain Model system, or the `Sink` to be specific. The `Sink` in Figure 4.10 is now portrayed in top-down view (hence the rounded corners) instead of the previously seen side view (e.g., Figure 4.1).

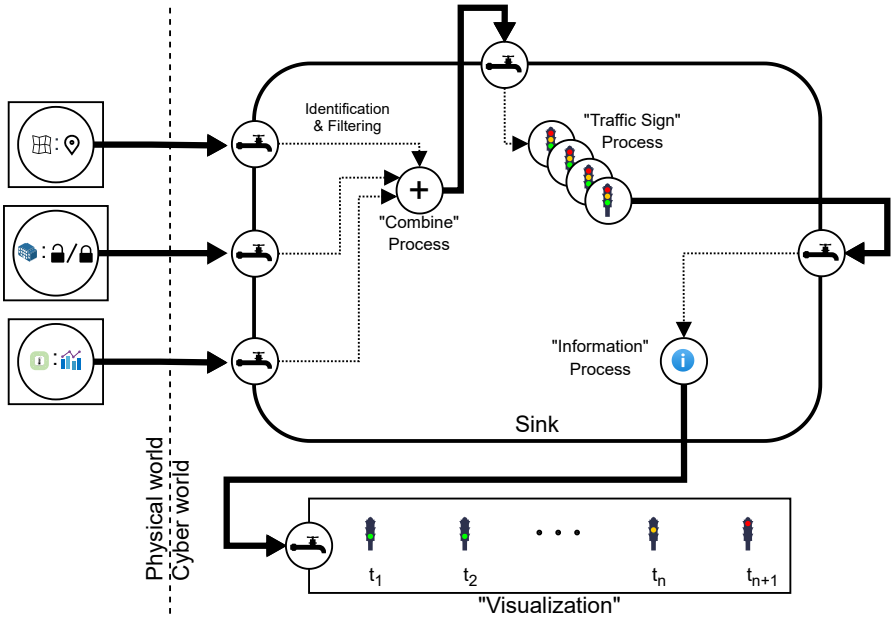


Figure 4.10: Structure of an example system.

In the Figure 4.10, the dotted arrows inside the system represents a data `stream`. The connected arrows also indicate the completed task of identification & filtering. The head of an arrow marks the acquisition step for the algorithms performing a processing task. Each processing task in the sink is an application specific variation point, encapsulated in a circle. In the context of Faucet-Sink-Drain Model, the circle indicates the `sieve` component. Processing tasks can be further connected together in order to create the feedback mechanism (Publication V). These are indicated by wide arrows, which remarks similarity to the initial data flow of the raw data sources.

The first processing task is a combiner process that gathers the data from the sources. The combiner can be a simple algorithm that selects the latest values from designated variables and streams. The output of the process would be forwarded to the next processing task. For the sake of a succinct presentation, the next step is actually a group of processing tasks, denoted by a group of encircled traffic lights. The group is also an abstraction that represents the simplified steps of turning the raw data into the information. Lastly, the results from the traffic sign process can be exported from the system through the information process task. This task (i.e., sieve) is an endpoint that can provide exportable `drain` interface to serve the selected data to various users and/or applications. In this case it is used to provide the data for an external visualization system. This visualization system

could be another Faucet-Sink-Drain Model system as is hinted by the `faucet` icon on the left side of the rectangle.

As a summary, the Faucet-Sink-Drain Model provides a highly adaptable set of functionalities that can be easily extended. In principle, all processing tasks (i.e., the variation points inside the sink) and the communication between those provide a similar generic interface. Therefore, any system is more simple to design and implement, regardless of the actual operations used. The application specific variation to processing tasks is "just a simple matter" of replacing the contents inside the variation points. From the software development point of view, this can present large potential for reusable components, as well as for the reusability of the data. Further, fixing errors in the software (i.e., bugs) may become easier, as the changes can be targeted to a specific encapsulated processing task instead of the whole system.

4.4 CONCLUSION OF THE EMPIRICAL STUDY

The Faucet-Sink-Drain Model and MDS component framework were iteratively improved throughout the research process. The first iteration was the recognition of the pipelined data processing model thru the metaphor of a streaming water apparatus, consisting of faucets, sinks, and drains. With the initial recognition of the model, it was possible to gain understanding in achievable capabilities. This understanding enabled to try the idea in a problem domain, which ended up to be in data source management. As a result, the reference architecture of MDS component framework (Publication V) was designed.

The next logical step was to prove that the model could be utilized in the creation of a framework. This was demonstrated by the realization of the prototype system, described in Section 4.3. The prototype system is an empirical result of the iterative search process. The prototype system also serves its purpose as a testing platform for the model. In the subsequent iterations of the search process, the model was defined more precisely and its theoretical implications were better identified (when compared with Publication V and Publication VI). The general purpose data processing model was described with the examples of prospective use cases (Section 4.2). Ideally, these use cases represent a subset of capabilities that can be implemented with the model.

Faucet-Sink-Drain Model was built upon on the idea of controlling the data processing cycle more systematically to aid software development. The best practices in SE have shown that there are good architectural choices and design patterns that can improve the quality of software, for example, in terms of higher reuse or better maintainability. The main point of the model is how the model treats all data sources as simply and generically as possible. The generic data source management of the model is the key to improving the reuse of source code as well as providing guidelines for implementation of new data

source processing. The defined model of data processing is seen as a distinctively separable component from the application logic, which supports general purpose implementations. Furthermore, the quality, reusability and maintainability of the data itself is increased due to the improved interpretation and tracing process enabled by the system.

The prototype system demonstrates that it follows the data processing tasks described in Section 3.1.2. It also meets the quality parameters defined in Section 3.1.1 in the following ways:

Accountability The traceable operations make it possible to store and retrieve the whole data processing log to each data stream. With this feature, the user would be able to trace the conducted operations all the way back to the original source.

Compatibility and Interoperability The architecture of the model and the MDS component framework encourages for a standardized approach to data processing: all data is/are equal, and therefore all data is being handled in the same way. The user would be able to rely on and utilize similar software constructs in different use cases. The standardized approach also improves the interoperability when communicating between other systems compatible with Faucet-Sink-Drain Model.

Reusability and Reproducibility With the aid of the accountable data, the system is able to provide consistent results with the same data and processing functions. The user would be able to rely on having reusable results in different environments, and is also able to share results consistently with other users and/or systems.

Reliability and Availability The storage model of the system has support for flexible and always up-to-date access to the original and processed data. The user would be able to find, access and manipulate any data stored in the system.

However, the system requires more focus on the development and implementation of new features in future research. For example, the quality parameters of compatibility and interoperability tested partially as the necessary features for a concrete client-server test system have not yet been implemented. The design of the prototype system supports compatibility, and the initial testing with the prototype system has indicated promising results. In theory, the system should be capable of fulfilling the quality parameters once the implementation is available. It might be worth to review the FAIR principles (Wilkinson *et al.*, 2016) which was referred to in Section 2.2.1. Further discussion about additional considerations and implications can be found in Chapter 5.

5

DISCUSSION

This chapter discusses the results of the research in wider scope to offer views for possible uses and further research. It also considers how well the research goal introduced in Section 1.2 was achieved.

5.1 REVISITING THE RESEARCH QUESTIONS

This section revisits the RQs mentioned in Section 1.2, which are duplicated below:

RQ #1 How to manage the data quality & data processing systematically?

RQ #2 How to design and perform data processing cycle?

RQ #3 How to comprise & execute the approach of generic model in a software application?

The RQ #1 can be answered with the knowledge gained from the background research of data management and similar approaches. The conclusion of the background research (see Section 2.4) was that there were no comprehensive solutions available that provide maintainable and systematic ways for managing the data. It was also realized that the typical information systems operate in five key areas, and should be benchmarked against a reference set of quality parameters (i.e., ISO/IEC 25010:2011, 2011; ISO/IEC 25012:2008, 2008). Both of these were described earlier in Section 3.1. The answer to RQ #2 is the design of a pipelined data processing model, which assigns each step with certain roles and tasks. The model in question was described in Section 4.1. The result continues the traditional practice of creating design patterns (e.g., Gamma *et al.*, 1994; Sommerville, 2016), providing a structurally new method for repeatable data processing in software. This enables more clarified and comprehensible system development. Lastly, the RQ #3 is a demonstration with a software prototype system, which integrates the requirements of the former questions. The prototype system was described in Section 4.3. The work in this part contributes significantly to practice rather than literature.

The idea of Faucet-Sink-Drain Model can be applied to various data processing scenarios because of its generic approach. It was concluded

to conduct experiments to improve data management because the model also had features that allowed it to operate with different types of data sources, and therefore would make it of the most fruitful application areas. Further study (see Chapter 2 and Section 2.1) also revealed that there was a gap when combining data quality, software quality, and systematic data management together. The model treats data as generic, simple enough, and easily reusable content. This can promote more efficient access to any data. With help of managing and tracking the data sources, the model has prospectives of providing reusable data in reusable software.

As one of the key results of the thesis, the generic approach was utilized in a data management application, where it was gained a better grasp of data sources in five key areas: acquisition, storage, identification & filtering, processing, and with certain limitations, visualization. These steps represent a data processing cycle, in which each step is well defined. The formal definition helps in the utilization of any data related needs in a software application. For example, it should be clear for the user to identify the entry point of the data, as well as the necessary activities for that particular step. The outcome of each processing cycle can be utilized in subsequent processing cycles again, as many times as necessary.

The Faucet-Sink-Drain Model is capable of solving problems in different scenarios other than data management. Different systems that are compatible with Faucet-Sink-Drain Model can find compatible parts that can be reused. From the SE point of view, this supports the reusability of the source code, and provides a suitable design pattern to be used in software applications dealing with data.

5.2 CONTRIBUTIONS OF THE THESIS

The invariable main contribution of this thesis is the design and development of the Faucet-Sink-Drain Model and the demonstration of the model in the form of MDS component framework. In this section, a selection of benefits leading toward the main contribution, are discussed.

The model defines a standardized approach for data processing by specifying how the data needs to be utilized, which enables systematical data management. By utilizing the systematical approach, it provides reusability at source code level (by offering patterns) as well as reusability of the data. Increased reusability improves the logical use of the data. Data managed by the model can be accompanied by metadata which stores the complete, traceable processing history, providing accountability for the managed data. With accountability, it is easier to determine the original source and understand the complete processing cycle, achieving in a more coherent procedure to modify any single step in the data processing.

From the data management point of view, knowing that the data is accountable can have benefits such as reuse of data, understanding how the resulting data (or information) was obtained, and make use of previously utilized methods with a different data set. In the scope of the model, the accountability of data can be observed on a few logical formations. A single sink formation contains everything within one sink. This representation is the one that was described in this thesis. In this formation all data, data cycles, processing, and history is accessible from a specific sink. In a sequential sink formation, the configuration goes into a strictly sequential mode with multiple sinks having a single cycle each. This is logically identical to the previous one, but there can be only one operation per sink. History of each step remains accessible as long as intermediary sinks are not removed. An unmanaged formation can be expressed as a sink with just one, very large cycle. Everything happens inside single processing cycle, thus only the state of the initial and the final conditions are observed.

Reusability of the source code related to the use case 1 was studied by manually inspecting the prototype system and the MDS component framework. The analysis can be seen in Table 5.1. The analysis includes a selected quantitative and qualitative metrics in the following assessments: source code level analysis is conducted by a tool named `cloc`, and a reusability analysis by examining the source code components that are prospective candidates to be incorporated into the future versions of the framework. It turned out that a remarkable part of the prototype system total code-base could be reusable in the framework. About 13 % of the code could easily integrate into the framework. Additionally, approximately 76 % is potentially reusable in the framework after refactoring the code. The remaining code is application specific that cannot be transferred.

Table 5.1: Analysis of the source code.

Metric	Framework	Prototype System
Lines of code (total)	196	993
Lines of blanks & comments	97	271
Number of files	11	16
Distinctly reusable		13 %
Candidates for reuse		76 %

Most of the framework code consists of interface definitions or comments, thus the greater part of the features are implemented by the prototype system. However, the forthcoming iterations could adopt the most useful and productive features directly into the framework. Until this happens, the performance in computational capacity might actually be lower when compared with conventional approaches, such

as those following procedural programming. Performance in development capacity can exhibit the same behavior as even the most simple components doing the processing work must be implemented from the scratch (e.g., faucets, sieves, filter rules et al.). However, any single component is easily reusable in the forthcoming development efforts, and can be incorporated into the tool box of the prototype system. In addition to the wider selection of tools, a good generic solution could be made as configurable recipes (Publication V) for the users to adopt in their own programs. One recipe is a well-defined series of processing cycles, that is intended to improve the productivity. The hypothetical examples of such recipes could be "select a data stream from data source X, and combine it with my own data," or "perform outlier detection in data stream Y."

Related to the reusability of data, a set of quality parameters was introduced in Section 3.1.1. Basically, it is a (ISO/IEC 25012:2008 data quality model) draft for data quality and data management extension. The draft includes definitions for characteristics that should be featured in systematically operated data applications. The characteristics were chosen in the way that a balanced workload could be maintained while allowing solutions to improve data quality issues. The accountability, interoperability, reusability, and reliability of data may prove useful in future as the amount of automated decisions based on data keeps growing.

5.3 EVALUATION OF THE CONSTRUCTS

This section reflects on the validity of the research in terms of the selected research goals. The evaluation of the model relies on the conclusions drawn from prior knowledge of related studies and the empirical evidence from the implementation and testing of the prototype system.

RESOLUTION TO RO AND RQS Previously, the individual RQs were discussed in Section 5.1. To return to the RO presented in Section 1.2, a response can be formulated according to what has been discovered.

RO There exists a generic model that enables the processing of data in any given software application.

Based on the findings and the answers discovered for the RQs, the Faucet-Sink-Drain Model has shown capabilities of satisfying this objective. It is generic enough to be adaptable to different requirements, and it can be integrated directly into a software application (see Section 4.3). The model is able to provide data processing operations according to the quality parameters as indicated in Section 4.4. A subset of the model features are tested with the prototype system (i.e., MDS component framework).

DESIGN PRINCIPLES OF THE MODEL The structure and architecture of the model is presented in Section 4.1. The architecture of Faucet-Sink-Drain Model is designed to support general purpose implementations by being generic enough to be utilized in different use cases. The focus areas are in improvement of systematic data processing approaches, and in data and software quality. The systematic control of data processing described by Faucet-Sink-Drain Model aids software development. This is the key in improving the reuse of source code as well as providing guidelines for the implementation of new data processing tools. Furthermore, the quality, reusability and maintainability of the data itself is increased due to the improved interpretation and tracing process enabled by the system.

APPEARANCE OF THE MODEL IN USE CASES This theme is dealt with in Section 4.2 with three increasingly more demanding use cases. The first of them is the most basic one where the Faucet-Sink-Drain Model is utilized in accessing and manipulating a local CSV file. This is also a concrete example of the Faucet-Sink-Drain Model as this case has been demonstrated by the developed prototype system.

The second case is a small increment to the previous use case, but it offers great potential for extended system features. In this case, the system utilizes data from an external data source. This is a theoretical example of the model as the current prototype system does not implement the accessing of remote data sources. However, in theory, the user could simply configure their own system to retrieve the interesting data from another system. Then, both of the Faucet-Sink-Drain Model compatible systems would negotiate and handle the mode of data transfer autonomously, perhaps even without the need to write additional code. As a consequence, the user would have more time to focus on implementing the functionalities regarding the data processing, thus increasing the productivity of the user.

The third case described more complex system operating with multiple data sources. Similar to the second case, there is no current implementation, but the value of this case is on the viewpoint of successfully applying the Faucet-Sink-Drain Model retroactively to an existing system. Thus, it indicates the model is capable and generic enough for a wide range of applications. This case was explained in more detail in Section 4.3.3.

APPLICATION PRACTICES OF THE MODEL This topic is considered in Section 4.3, and it presents in more detail about how the Faucet-Sink-Drain Model is utilized in an application. The prototype implements the model as a framework to provide the basic features. The framework is the generic part that may remain the same in different applications. Specific part of the application is defined by the use case specific requirements. The framework is attached to the application

by the means of variation points (e.g., inheritance, parameterization, interfaces) that are one of the basic features in the software development. Specific part of the prototype application will then declare how it utilizes the generic and reusable fragments of the model (mainly five key components; faucets, streams, sinks, sieves, and drains) to complement missing framework features with various application specific implementations. The remaining unspecific part is uncontrollable which can make the discovery of the model more complicated. Generalization cannot reach this part, but it is nevertheless important to identify the areas which are beyond the scope of the model.

The generic and specific parts were distinctively depicted on Figure 4.9. The organization of components inside processing cycles (i.e., clusters) remain similar, while the necessary parts of components are changed to meet application specific needs. The variation points of the model were shown from another viewpoint in Figure 4.10 where each full processing cycle is depicted as a sequence of variable operations.

IMPROVEMENT OF THE QUALITY ASPECTS Improvement of the quality aspects of a software and data were in a central role as the model was being implemented. Those were discussed in more detail in Section 3.1.1, and Section 4.4 compiles how the quality aspects were improved by the Faucet-Sink-Drain Model.

The main reasons for using the model is in its approach of data processing: all data are "created equal," and therefore all data is being handled in the same way. The data processing can be splitted and pipelined over several processing cycles instead of writing a single monolithic black box function. While the model allows implementation of such "large functions," the initial idea was to prefer more (reusable) processing cycles (per data point) and less work per cycle. Each processing cycle has identical and well-formed steps which can give predictable and reproducible output based on the input and the selected processing functions. From this perspective, the processing functions of the model system can be regarded stateless and pure. The inevitable drawback of this approach is the increased processing overhead, thus the pros and cons must be carefully taken into account when it comes down to the selection of the tools for a certain project.

The model can accept all kinds of data, and it can be utilized in different scenarios and use cases as shown above. Applications building upon the model can steer and encourage in selecting approaches that enable reusable general purpose constructs to be utilized. Reusability can appear at source code level as well as in the creation and organization of the processing cycles. Reusability can also appear in the utilization of data. The model aims for easier control of utilizing any existing data, and allows the user to indicate what kind of data should be retained for further use. Additionally, it is possible to trace back in the processing history of (any) data point, and see what partic-

ular steps have been carried out. The aspects discussed can return increased value in the form of more efficient software development.

SELF-ASSESSMENT One of the strongest points of the model is the ability to retrieve and manage a lot of similarly structured data effectively from the developer's point of view. However, the users (of the system) will be required to have good insight into the data so that they can reach the desired results. It can offer a "code once; process multiple times" approach which saves effort with a high level of reusable structures. The data from each different source should be well defined, which should aid in the ability of the system to handle laborious and repetitive low-level database routines. Additionally, if required, the system would be able to create the logical snapshots of the current work so it would be easy to come back to the same point with a) new data or b) different algorithms. All this allows us to concentrate on conducting more experiments with the data instead of the need to dilute the focus by building and tweaking the system.

Regarding the disadvantages, there are some notable points to consider. One, the sequence of operations may form very long chains, depending on the amount and complexity of the feedback loops being utilized. Another consideration is that the selector creation can be rather prone to mistakes as no sanity checking is done. As it stands, the selectors and processors need to be written as code since there are no other tools for creating them. Ideally, the user would have a helper application to make the rule writing easier. This could alleviate some issues with the sanity checking of rules. Similar issues also manifest in the current implementation of the selection and processing procedure (especially Listing 4.2), so improvement in that regard is required for reaching higher maturity level.

Is this model an answer to any and all data (source) management needs? Will the applications implementing it, immediately become better? Evidently, there is still a lot to be done, and the model is not a magical silver bullet. It should be considered more like the beginning of a paradigm for tackling the issues arising in data handling. Also, it is probably not a complete back-end data solution as is, but is better viewed as a specialized component in the application architecture. In this way, the model can be implemented to perform the single logical task of data source management. This would partially satisfy the Unix philosophy (e.g., Gancarz, 1995; Raymond, 2003) of software development where a program should be made to "do one task and to do it well." It appears that the size of systems has grown larger since the term was coined, but that does not mean that everything must be built into one monolithic system. Thus, with smart architectural design choices, the responsibilities of a certain application feature(s) can be partitioned into smaller, simple, and manageable software artifacts. Of course, doing just one task would contradict with many of the

previous arguments in this thesis, but perhaps it can be solved by rephrasing the sentence to "do one *processing cycle* and to do it well."

5.4 THREATS TO VALIDITY

This section reflects on the validity of the research in terms of the chosen research methodology, DSRM, proposed by Peffers *et al.* (2007). DSRM defines a process model consisting of six research activities. The outcome of the activities can be used as a way to confirm validity. The mapping of the activities to the chapters of this thesis is summarized in the following listing:

Chapter 1 Activity 1: Problem Identification and Motivation
Activity 2: Objectives of the Solution

Chapter 2 Activity 1: Problem Identification and Motivation

Chapter 3 Activity 2: Objectives of the Solution

Chapter 4 Activity 3: Design and Development
Activity 4: Demonstration

Chapter 5 Activity 5: Evaluation

The first two activities provide prior knowledge and boundaries for the research. The background study conducted and reported on Chapter 2 deals with Activity 1. The results and relevance of this activity can be validated with the help of the literature review results described in the chapter. The second activity described in Chapter 3 provides the necessary theory from the prior work and knowledge of the thesis author. Activities 3 and 4 can be earmarked for Section 4.1 and Section 4.3, respectively. Chapter 4 in its entirety also fulfills the first guideline of DS research (Hevner *et al.*, 2004): design as an artifact. The fifth activity, evaluation and comparison of the results, is conducted in the current chapter, Chapter 5. The sixth activity, communication, is not specifically placed in any single chapter or section. However, communication was conducted during the research by the means of publications, especially Publication V and Publication VI. The feedback from these papers was utilized to guide the research. Furthermore, this thesis as a whole can be seen as fulfilling the communication aspect.

Regarding the potential risks of the research, the model and architecture was put to the test over three use cases Section 4.2 and a single prototype system Section 4.3. The generic nature of the model can make it hard to affix to a concrete example, but a reasonable amount of effort was used to make the cases representative enough. For example, the process iteration of DSRM enabled to revise and adapt use cases based on the intermediary evaluation of the model (see Figure 1.2 outlining the chosen research process). The selected use cases cover

typical data processing needs, explaining the intended usage and therefore the most appropriate setting for the system. The described prototype system was shown to perform consistent with the conventional methods in the first case, and according to the implementation level of the prototype system, promising results in the others.

5.5 FUTURE WORK

This section gives indications of where to continue the research from here. The previous sections in this chapter have already included a few suggestions for improvement in their respective contexts, so those are not repeated unnecessarily in this section.

The basic concept of the model (i.e., Faucet-Sink-Drain Model) has remained largely similar throughout the research, but there are always avenues for improvement. The most obvious points for improvement are in the MDS component framework and applications building upon it, as they are still gaining the necessary maturity for large-scale utilization. The architecture could benefit from a re-evaluation and fine-tuning of the organization of its internal components. In hindsight, the change in visualizations mentioned along Table 4.1 was not fully carried out, as the final step of the data processing could be *consumed* by an automated machine-to-machine system. Therefore, the terminology in the visualization step should be replaced with more relevant terms. Additionally, it could prove an interesting experiment to utilize the Apache Flink ecosystem to implement the current MDS component framework logic as a community package for the Apache Flink. Either way, there have been discussions on publishing the source codes of the prototype on a public platform.

The applicability of the model should be documented in more scenarios to gain more understanding and feedback on the overall system. This includes, among others, the full evaluation of FAIRness of the Faucet-Sink-Drain Model. The prototype system has shown that it can handle a few data streams and data process cycles at the same time, but it will be interesting to see also how effectively it can handle very large amounts of data and data streams simultaneously.

THEORETICAL USE CASES While the MDS component framework is specialized in tackling issues of data source management, the model has the flexibility to make it useful in various other approaches. In the following, a few theoretical scenarios are discussed. These are based on the potential improvement topics discovered during the research on the model and the prototypes:

Multi-sink Depending on the case, the user could assign many data sinks for different purposes. For example, the user is able to create different data sink environments for the tasks they are performing, the user would be able to

make clones (duplicating data), or create the read-only mirrors of the data sink.

Pooling — Data sinks can be chained so that the flow of information is additive and scaleable through an increased level of parallelism. The idea bases on the flow of information and data builds up, forming bigger streams of data. The bigger streams can be formed from the valuable information, or even from the discarded data.

Accessibility — In this approach, the physical location of the sink can be freely chosen. For example, data sinks could be made accessible from everywhere through Internet services, or by keeping them on a portable mass storage for mobility or security purposes.

Ownership — The bearer (or owner) of the sink can control what can be shared to third parties. With the help of accessible data sinks, user-related data such as the customer information of an online shopping service could be stored on personal data sinks instead of storing them on each and every web service independently.

The potential presented by the model may be a precursor to a new kind of decentralized service data infrastructure. For an example, it could enhance a signature service which performs a digital signing of documents among its users. The data can be distributed across different service providers. However, the signature service would not store the data in their own servers. Instead, it would act as a gateway to the users to transmit signature requests to each other and to provide the certification of the authenticity of the signatures. The service would not need to host a large repository for the storage of the payload data. This way, any service could specialize further, and would not need to maintain features deemed unnecessary for their core business.

CONCLUDING REMARKS Reflecting on the implementation level of the MDS component framework, there are some features lacking in the current prototype system. However, this is to be expected as the intention of the research artifact was not a fully featured product for a regular system developer, but to indicate the feasibility of the selected approach. The research has shown that the idea is feasible on the proof-of-concept level. While the concrete realizations are less ideal than originally envisioned, the theoretical implications of the research provide open pathways for introducing more features, testing, and development efforts. As it stands, the primary effort needs to be focused on the creation of reusable components and recipes so that the system would have better tools and well-defined solution patterns available. The wider tool selection can increase level of automation for the development process.

6

SUMMARY

This thesis studied how data management goals can be achieved in software development from the perspective of data sources. The background research indicated that there was a gap in data management in combining data quality, software quality, and a systematic approach together. It was concluded that a generic model which can operate with the data processing of data sources efficiently would improve the situation. A generic data management model was developed utilizing these findings, along with good architectural design points and software standards.

The Faucet-Sink-Drain Model was built upon on the idea of controlling the data processing cycle more systematically to aid software development. The main point of the model is how the model treats all data sources as simply and generically as possible. The generic data source management of the model is the key to improving the reuse of source code as well as providing guidelines for the implementation of new data source processing tools. Furthermore, the quality of data, on such parameters as accountability, compatibility, reusability, and reliability, are increased due to the improved interpretation and tracing process enabled by the system. The data source processing architecture was designed to support general purpose implementations. The aforementioned aspects of the model were tested in the MDS component framework system.

MDS is an architecture that covers the creation of data, includes the search, selection, and combination of any input data, and finishes with the outcome of the processed data. Reflecting on the implementation level of the MDS component framework, there are several improvement paths available both on the theoretical and practical side. However, this was an expected outcome as the intention of the research artifact was not to be a fully featured product, but to indicate the feasibility of the selected approach.

The research has shown that the idea is feasible, thus more features, testing, and development efforts are good avenues for future research. The adoption of the generic data processing model is practicable for various software applications. As a result, the emphasis of data processing development efforts could shift more toward the creation of useful content and information instead of the handling of individual

data points. The Faucet-Sink-Drain Model can potentially aid in efforts to improve data accessibility, reusability, and ownership by means of a data processing infrastructure which allows a greater decentralization of service data.

The experiments with prototypes indicate that the proposed Faucet-Sink-Drain Model can be implemented in a software application. The theoretical background studies support the notion that the problem described in this thesis is relevant, and that the choices in the designed architecture can improve the quality of software. Therefore, it can be concluded that there exists a generic and systematically manageable data processing model, which can improve the quality of both software and data.

REFERENCES

D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "Scalable semantic web data management using vertical partitioning," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, ser. VLDB '07, Vienna, Austria: VLDB Endowment, 2007, pp. 411–422, ISBN: 978-1-59593-649-3. [Online]. Available: http://dl.acm.org/citation.cfm?id=1325851.1325900.

R. L. Ackoff, "From Data to Wisdom," *Journal of Applied Systems Analysis*, vol. 16, no. 1, pp. 3–9, 1989.

D. Agrawal, A. El Abbadi, S. Antony, and S. Das, "Data management challenges in cloud computing infrastructures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5999 LNCS, pp. 1–10, 2010. DOI: 10.1007/978-3-642-12038-1_1.

M. A. Babar and H. Zhang, "Systematic literature reviews in software engineering: Preliminary results from interviews with researchers," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 346–355.

Y. Baolong, W. Hong, and Z. Haodong, "Research and application of data management based on Data Management Maturity Model (DMM)," in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, ser. ICMLC 2018, Macau, China: ACM, 2018, pp. 157–160, ISBN: 978-1-4503-6353-2. DOI: 10.1145/3195106.3195177.

M. I. Bellgard, "ERDMAS: An exemplar-driven institutional research data management and analysis strategy," *International Journal of Information Management*, vol. 50, pp. 337–340, 2020, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2019.08.009.

C. Bisdikian, B. Mitschang, D. Pedreschi, V. S. Tseng, and C. Bettini, "Challenges for mobile data management in the era of cloud and social computing," in *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management*, vol. 1, Jun. 2011, p. 6. DOI: 10.1109/MDM.2011.104.

F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, A System of Patterns*. Wiley, 2013, ISBN: 9781118725269.

C. Campbell, *Top Five Differences between Data Lakes and Data Warehouses*, 2015. [Online]. Available: https://www.bluegranite.com/blog/bid/402596/top-five-differences-between-data-lakes-and-data-warehouses/ (Accessed: 21 Jan. 2021).

D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik, "Monitoring streams: A new class of data management applications," in *Proceedings of the 28th International Conference on Very Large Data Bases*, ser. VLDB '02, Hong Kong, China: VLDB Endowment, 2002, pp. 215–226. [Online]. Available: http://dl.acm.org/citation.cfm?id=1287369.1287389.

Y.-M. Chen and T.-H. Tsao, "A structured methodology for implementing engineering data management," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 4, pp. 275–296, 1998, ISSN: 0736-5845. DOI: 10.1016/S0736-5845(98)00013-1.

Y. Cheng, X. Zhang, P. Wang, L. Zha, D. Hou, Y. Qi, and C. Ma, "Data management challenges and event index technologies in high energy physics," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 54, no. 2, pp. 258–266, 2017. DOI: 10.7544/issn1000-1239.2017.20160939.

P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford, *Documenting Software Architectures: Views and Beyond*, ser. SEI Series in Software Engineering. Pearson Education, 2010, ISBN: 9780132488594.

CMMI Institute, *CMMI Institute – Data Management Maturity (DMM)*, 2019. [Online]. Available: https://www.cmmiinstitute.com/data-management-maturity (Accessed: 28 Oct. 2019).

Deloitte Consulting LLP, *How Data Lakes Fit Into a Modern Data Architecture*, 2018. [Online]. Available: https://deloitte.wsj.com/cio/2018/09/05/how-data-lakes-fit-into-a-modern-data-architecture/ (Accessed: 21 Jan. 2021).

W. Dong, X. Zhang, and B. Jiang, "Service-based distributed data management and application in China Digital Ocean," 2010, pp. 425–428. DOI: 10.1109/IGARSS.2010.5649883.

Y. C. Du, J. Hongli, G. Lili, and L. Xin, "Comparative analysis of data management system," in *Proceedings of The 2016 6th International Conference on Machinery, Materials, Environment, Biotechnology and Computer (MMEBC)*, Zhang, L and Xu, D, Ed., ser. AER-Advances in Engineering Research, vol. 88, Atlantis Press, 2016, 919–923, ISBN: 978-94-6252-210-7. DOI: 10.2991/mmebc-16.2016.192.

Elasticsearch B.V., *Elasticsearch: The Official Distributed Search & Analytics Engine | Elastic*, 2021. [Online]. Available: https://www.elastic.co/elasticsearch/ (Accessed: 21 Jan. 2021).

T. Z. Emara and J. Z. Huang, "A distributed data management system to support large-scale data analysis," *Journal of Systems and Software*, vol. 148, pp. 105–115, 2019, ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.11.007.

Environment Agency UK, "Quality and Performance Standards for Environmental Data Management Software," Environment Agency United Kingdom, Rotherham, UK, Guidance, Dec. 2017. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/664289/LIT_5787.pdf.

Finnish Social Science Data Archive, *Data Management Guidelines*, urn:nbn:fi:fsd:V-201504200002, Feb. 2015. [Online]. Available: https://www.fsd.tuni.fi/en/services/data-management-guidelines (Accessed: 16 May 2021).

M. J. Franklin, "Mobile data management – a dozen years later," in *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management*, ser. MDM '11, vol. 1, USA: IEEE Computer Society, Jun. 2011, p. 3, ISBN: 9780769544366. DOI: 10.1109/MDM.2011.102.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 32nd, ser. Addison-Wesley Professional Computing Series. Pearson Education, 1994, ISBN: 0-201-63361-2.

M. Gancarz, *The UNIX philosophy*. Digital Press, 1995, ISBN: 1555581234.

Gazali, S. Kaur, and I. Singh, "Artificial intelligence based clinical data management systems: A review," *Informatics in Medicine Unlocked*, vol. 9, pp. 219–229, 2017, ISSN: 2352-9148. DOI: 10.1016/j.imu.2017.09.003.

D. Ghoshal and L. Ramakrishnan, "FRIEDA: Flexible robust intelligent elastic data management in cloud environments," 2012, pp. 1096–1105. DOI: 10.1109/SC.Companion.2012.132.

J. Greenfield and K. Short, "Software factories: Assembling applications with patterns, models, frameworks and tools," in *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, ser. OOPSLA '03, Anaheim, CA, USA: ACM, 2003, pp. 16–27, ISBN: 1-58113-751-6. DOI: 10.1145/949344.949348.

A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, May 2004.

J. W. Horch, *Practical Guide to Software Quality Management*, ser. Artech House computing library. Artech House, 2003, ISBN: 9781580536042.

S. Hossmann, A. G. Haynes, A. Spoerri, I. D. Diatta, B. Aboubacar, M. Egger, F. Rintelen, and S. Trelle, "Data management of clinical trials during an outbreak of ebola virus disease," *Vaccine*, 2017, ISSN: 0264-410X. DOI: 10.1016/j.vaccine.2017.09.094.

W. H. Inmon, *Building the Data Warehouse*. New York, NY, USA: John Wiley & Sons, Inc., 1992, ISBN: 0471569607.

ISO/IEC 20546:2019, "Information technology – Big data – Overview and vocabulary," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 20546:2019, Feb. 2019. [Online]. Available: https://www.iso.org/standard/68305.html.

ISO/IEC 20547-3:2020, "Information technology – Big data reference architecture – Part 3: Reference architecture," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 20547-3:2020, Mar. 2020. [Online]. Available: https://www.iso.org/standard/71277.html.

ISO/IEC 25010:2011, "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 25010:2011, Mar. 2011. [Online]. Available: https://www.iso.org/standard/35733.html.

ISO/IEC 25012:2008, "Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 25012:2008, Dec. 2008. [Online]. Available: https://www.iso.org/standard/35736.html.

ISO/IEC 42010:2011, "Systems and software engineering – Architecture description," International Organization for Standardization, Geneva, CH, Standard ISO/IEC/IEEE 42010:2011, Dec. 2011. [Online]. Available: https://www.iso.org/standard/50508.html.

ISO/IEC TR 20547-2:2018, "Information technology – Big data reference architecture – Part 2: Use cases and derived requirements," International Organization for Standardization, Geneva, CH, Standard ISO/IEC TR 20547-2:2018, Jan. 2018. [Online]. Available: https://www.iso.org/standard/71276.html.

H. Jaakkola and B. Thalheim, "Trends and Future of Data Modelling," in *Information Modelling and Knowledge Bases XXXII*, ser. Frontiers in Artificial Intelligence and Applications. Netherlands: IOS Press, Feb. 2021.

V. Kantere, "A holistic framework for big scientific data management," in *2014 IEEE International Congress on Big Data*, Jun. 2014, pp. 220–226. DOI: 10.1109/BigData.Congress.2014.39.

——, "Datom: Towards modular data management," in *2015 IEEE International Conference on Information Reuse and Integration*, Aug. 2015, pp. 443–450. DOI: 10.1109/IRI.2015.74.

——, "Datom continued: Towards multi-objective optimization of data management entities," in *17th IEEE International Conference on Information Reuse and Integration, IRI 2016, Pittsburgh, PA, USA, July 28-30, 2016*, 2016, pp. 275–282. DOI: 10.1109/IRI.2016.44.

N. Khan, M. Alsaqer, H. Shah, G. Badsha, A. A. Abbasi, and S. Salehian, "The 10 Vs, issues and challenges of big data," in *Proceedings of the 2018 International Conference on Big Data and Education*, ser. ICBDE '18, Honolulu, HI, USA: ACM, 2018, pp. 52–56, ISBN: 978-1-4503-6358-7. DOI: 10.1145/3206157.3206166.

B. A. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009, Special Section – Most Cited Articles in 2002 and Regular Research Papers, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2008.09.009.

B. A. Kitchenham, D. Budgen, and O. P. Brereton, "Using mapping studies as the basis for further research – a participant-observer case study," *Information and Software Technology*, vol. 53, no. 6, pp. 638–651, 2011, Special Section: Best papers from the APSEC, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2010.12.011.

M. Kleppmann, *Designing data-intensive applications: the big ideas behind reliable, scalable, and maintainable systems*, 1st ed. O'Reilly, 2017, ISBN: 978-1-449-37332-0.

K. Koskimies and T. Mikkonen, *Ohjelmistoarkkitehtuurit*. Talentum, 2005.

D. Laney, *3-D Data Management: Controlling Data Volume, Velocity and Variety*, (Broken link as of 6 November 2020), Feb. 2001. [Online]. Available: https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

L. Li, H. Wang, J. Li, and H. Gao, "A survey of uncertain data management," *Frontiers of Computer Science*, vol. 14, no. 1, 162–190, Feb. 2020, ISSN: 2095-2228. DOI: 10.1007/s11704-017-7063-z.

M. Lloyd, "Performance standards and test procedures for environmental data management software," *Proceedings – 1st IMEKO TC-19 International Symposium on Measurement and Instrumentation for Environmental Monitoring*, 2007.

A. Magdy and M. F. Mokbel, "Towards a Microblogs Data Management System," in *2015 16th IEEE International Conference on Mobile Data Management*, vol. 1, Jun. 2015, pp. 271–278. DOI: 10.1109/MDM.2015.24.

B. Marr, *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read*, 2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read (Accessed: 6 Nov. 2020).

T. Miksa, J. Cardoso, and J. Borbinha, "Framing the scope of the common data model for machine-actionable data management plans," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018, pp. 2733–2742. DOI: 10.1109/BigData.2018.8622618.

MongoDB, Inc, *Aggregation Pipeline – MongoDB Manual*, 2019. [Online]. Available: https://docs.mongodb.com/manual/core/aggregation-pipeline/ (Accessed: 28 Oct. 2019).

T. T. Nguyen, "Improving software quality with programming patterns," Ph.D. dissertation, Iowa State University, 2013.

I. Nonaka and H. Takeuchi, *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995, ISBN: 9780195092691.

A. Patrizio, *IDC: Expect 175 zettabytes of data worldwide by 2025*, Dec. 2018. [Online]. Available: http://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025 (Accessed: 23 Oct. 2021).

K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. DOI: 10.2753/MIS0742-1222240302.

K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2015.03.007.

S. Plantikow, K. Peter, M. Högqvist, C. Grimme, and A. Papaspyrou, "Generalizing the data management of three community grids," *Future Generation Computer Systems*, vol. 25, no. 3, pp. 281–289, 2009. DOI: 10.1016/j.future.2008.05.001.

L. Pophal, "The State of Data Management," *ECONTENT*, vol. 42, no. 1, 28–30, 2019, ISSN: 1525-2531.

L. Qiao and W. Liu, "Agile manufacturing data management," *Key Engineering Materials*, vol. 407-408, pp. 189–193, 2009. DOI: 10.4028/www.scientific.net/KEM.407-408.189.

I. Rafique, P. Lew, M. Q. Abbasi, and Z. Li, "Information quality evaluation framework: Extending ISO 25012 data quality model," *World Academy of Science, Engineering and Technology*, vol. 65, pp. 523–528, 2012.

P. Rantanen, J. Mäkivaara, M. Saari, P. Sillberg, and H. Jaakkola, "Utilizing cost-effective NB-IoT-based sensors for detecting water temperature and flow," in *2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES)*, 2021, pp. 165–170. DOI: 10.1109/INES52918.2021.9512896.

E. S. Raymond, *The art of Unix programming*. Pearson Education, 2003, ISBN: 0131429019.

M. Ridley and C. Stoker, "Data management tools," *Bridging the Gap: Meeting the World's Water and Environmental Resources Challenges – Proceedings of the World Water and Environmental Resources Congress 2001*, vol. 111, 2004. DOI: 10.1061/40569(2001)52.

J. Rowley, "The wisdom hierarchy: representations of the DIKW hierarchy," *Journal of Information Science*, vol. 33, no. 2, pp. 163–180, 2007. DOI: 10.1177/0165551506070706.

P. Sillberg, J. Raitaniemi, P. Rantanen, J. Soini, and J. Leppäniemi, "Flexibly managed user interfaces for mobile applications," in *Proceedings of the IADIS International Conferences Web Based Communities 2011, Collaborative Technologies 2011 and Internet Applications and Research 2011, Rome, Italy, 20-26 July, 2011*, IADIS International Association for Development of the Information Society, 2011, pp. 151–159, ISBN: 978-972-8939-40-3.

P. Sillberg, P. Rantanen, and J. Soini, "A content based tool for searching, connecting and combining digital information – case: Smart photo service," in *Proceedings of the 16th International Multiconference Information Society, IS 2013, Volume A, 7-11 October 2013, Ljubljana, Slovenia*, 2013, pp. 249–252.

S. Soares, *The IBM Data Governance Unified Process: Driving Business Value with IBM Software and Best Practices*. MC Press, LLC, 2010, ISBN: 978-1-5834-7360-3.

I. Sommerville, *Software Engineering*, 10th ed., ser. Always learning. Pearson, 2016, ISBN: 9780133943030.

Statista, *Total data volume worldwide 2010-2024*, 2020. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/ (Accessed: 6 Nov. 2020).

R. Taylor, N. Medvidovic, and E. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. Wiley, 2009, ISBN: 9780470167748.

The Apache Software Foundation, *Apache Spark – Unified Analytics Engine for Big Data*, 2018. [Online]. Available: https://spark.apache.org (Accessed: 28 Oct. 2019).

——, *Apache Flink – Stateful Computations over Data Streams*, 2021. [Online]. Available: https://flink.apache.org/ (Accessed: 21 Jan. 2021).

——, *Apache Lucene – Welcome to Apache Lucene*, 2021. [Online]. Available: https://lucene.apache.org/ (Accessed: 23 Oct. 2021).

B. M. Thuraisingham, *Handbook of Data Management, 1998 Edition*, 1st. Boca Raton, FL, USA: CRC Press, Inc., 1998, ISBN: 084939953X.

C. Veesommai, Y. Kiyoki, P. Sillberg, J. Soini, H. Jaakkola, and P. Chawakitchareon, "The rSPA process realization: The creation of river heavy metal evaluation index (rHMEI) by using dimensional subspace of heavy metal," *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, vol. 7, no. 3, 2016, ISSN: 2228-9860.

R. Vieira, F. Ferreira, J. Barateiro, and J. Borbinha, "Data management with risk management in engineering and science projects," *New Review of Information Networking*, vol. 19, no. 2, pp. 49–66, 2014. DOI: 10.1080/13614576.2014.918519.

M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, 2016.

Y. Xue-rong, L. Jia-hong, L. Ying, L. Feng, F. Xiang-jun, and W. Yuanlan, "Study of universal simulation data management system," in *Proceedings of the 2009 International Conference on Information Technology and Computer Science – Volume 01*, ser. ITCS '09, Washington, DC, USA: IEEE Computer Society, 2009, pp. 333–338, ISBN: 978-0-7695-3688-0. DOI: 10.1109/ITCS.2009.75.

Part II

ORIGINAL PUBLICATIONS

This part consists of the original publications which found the basis of this compilation thesis.

P1

PUBLICATION I

Utilizing Adaptive Software to Enhance Information Management
J. Soini, P. Sillberg, and J. Raitaniemi

International Journal of Computer, Electrical, Automation, Control and Information Engineering 6(12) 2012, pp. 1553–1558

Utilizing Adaptive Software to Enhance Information Management

J. Soini, P. Sillberg, J. Raitaniemi

***Abstract*—**The task of strategic information technology management is to focus on adapting technology to ensure competitiveness. A key factor for success in this sector is awareness and readiness to deploy new technologies and exploit the services they offer. Recently, the need for more flexible and dynamic user interfaces (UIs) has been recognized, especially in mobile applications. An ongoing research project (MOP), initiated by TUT in Finland, is looking at how mobile device UIs can be adapted for different needs and contexts. It focuses on examining the possibilities to develop adapter software for solving the challenges related to the UI and its flexibility in mobile devices. This approach has great potential for enhancing information transfer in mobile devices, and consequently for improving information management. The technology presented here could be one of the key emerging technologies in the information technology sector in relation to mobile devices and telecommunications.

***Keywords*—**Emerging technologies, Flexible user interfaces, Information management, Information technology, Mobile technology.

I. Introduction

INFORMATION sharing and transfer are necessary elements of knowledge management in organizations. Accessibility and access to the company's back-end system information, corporate data and process infrastructure, anytime and anywhere, is essential. There is a growing need for more efficient checking, updating, and viewing of important company information in real time [1],[2],[3]. Contemporary telecommunications and mobile technology offer many different opportunities for meeting this critical business need. Using intelligent mobile devices and the applications made for them, users are able to collaborate and share information and capture knowledge when and where they need it. However, expectations and requirements for information are changing, and there are recognized needs for more flexibility, dynamic and personally tailored features in relation to the usability of mobile devices [4],[5]. On the other hand, the increasing fragmentation related to mobile devices and also the types of information to be transmitted, place major challenges on the implementation of device independent dynamicity. Moreover, there are some technical restrictions and limitations related to mobile services, which add new challenges to the widespread adoption and diffusion of mobile applications [6] as well as data transformation [7].

As a result of this situation, the MOP research project (MOP - adaptive software services) was initiated in early 2011 by the Tampere University of Technology (TUT) [8] in Finland. The research deals with exploiting the opportunities created by current mobile technology to meet the more extensive and increasing needs of users. The main aim in the ongoing project is to study technologies that can enable context-based adaptive user interfaces (UI) for mobile devices, and the project is examining the distribution options for an interface that can be adapted according to the situation. In the research, the primary focus is on technologies that enable run-time implemented user interfaces that adapt easily and quickly. This kind of feature offers new opportunities for exploiting the usage of mobile devices in a more versatile way that at present. The research will also try to help solve the problems and challenges faced by software developers when trying to create software for multiple platforms and devices that may not be natively compatible with each other, while still providing similar user experience on all target devices.

The purpose of this research is to indicate new opportunities for exploiting mobile technology in corporate information management. It focuses on increasing the potential of using the existing resources of the company – mobile devices in this case – as well as improving the availability, updating and transferability of real-time information, one of the most important factors from the aspect of information quality in an organization's operations. The proposed approach shows an innovative way to apply existing mobile technologies for enhancing fluent and suitable communication and information sharing in an organization. Mobile applications and services already released for this flexible and dynamic UI environment are not very widely available. However, in the research field, there have been several other previous approaches to studying UIs dynamically, such as Jou [9], who presents a way to make the web browsing experience on mobile devices better by transcoding the web tables into more suitable forms. Jin et al. [10] present a UI framework for the dynamic updating of a web page during UI interaction on local networks and over the Internet. Moreover, Ye and Herbert [11] have presented generic software architecture for an adaptive user interface development system, which is based on a standard abstract UI description language (XUL), which is shared among different platforms. Rosenthal [12] proposed a special case of adapter software in what is called the WebKit hybrid (see also [13]). In

J. O. Soini is with the Software Engineering Department, Tampere University of Technology, Pori unit, 28100 Pori, Finland (phone: +358-40-8262890; fax: +358-2-627 2727; e-mail: jari.o.soini@tut.fi).

P. K. Sillberg is with the Software Engineering Department, Tampere University of Technology, Pori unit, 28100 Pori, Finland (phone: +358-40-8262745; fax: +358-2-627 2727; e-mail: pekka.sillberg@tut.fi).

J. Raitaniemi is Director (Mobile Solutions) of Acando Oy, Valtakatu 6, 28100 Pori, Finland (phone: +358-44-3434100; fax: +358-424-723220; e-mail: janne.raitaniemi@acando.com).

addition, there are some more technology-oriented studies on this topic, which have given examples of using XML-based UI language in simple applications [14],[15],[16],[17].

The structure of this paper is as follows: the next section gives a brief introduction to the impact of mobile technology and also the opportunities it provides as a part of information management. Section II also explains the relevance of the research topics, i.e., dynamic user interfaces and also evaluates potential ways of exploiting it. Section III describes the studied and applied mobile technologies and also, with the help of examples, presents the preliminary results of the study. Section IV describes the approach of using adapter software for retrieving UI embedded with content and even for retrieving software updates. Section V includes a discussion and evaluation of the study results and also some proposals for future research on this topic. Finally, Section VI summarizes the paper.

II. Opportunities and Challenges of Mobile Technology in Information Management

In general, the technology management function in an organization is meant for understanding the value of a certain technology for the organization. In relation to this, it can be stated that information technology (ICT) can have a remarkable strategic impact on organizations [18]. ICT offers huge opportunities to organizations, and if exploited effectively, it can create competitive advantages and also economic value for organizations [19],[20]. Today an inseparable part – and one of the most rapid growing industries - of information technology is mobile technology. It is obvious that this sector has a huge potential to succeed in emerging technology markets in the near future. Organizations are realizing more and more clearly the huge potential that mobile information technology can offer [21]. There is a lot of evidence for positive effects such as acceleration of data availability, real-time data, quality of information, and decision making [3],[22]. There are also many indicators of the strategic implications of mobile technology, like increasing internal communication and knowledge sharing as well as flexibility in coordination [5],[23],[24]. The capability of utilizing the latest mobile technology and also awareness of new ways of applying, adapting and utilizing it, is just one of the most important and significant parts of a company's strategic technology management.

In modern business, one of the key issues is ubiquitous access to the organization's back-end system information, company data, and process infrastructure. Enabling access to corporate information anywhere and anytime is essential. Mobile devices are also becoming more suitable for mobile data use and the amount of more sophisticated mobile applications is quickly increasing [25]. We can state that in principle advanced mobile technology facilitates the real-time collecting and sharing of data, and its storage directly in the company's information systems, so that the updated data become available to everyone. However, in practice, there are several issues that restrict the fluent and seamless functionality in the kind of situation described above. Fragmentation could be an apposite term to describe the current situation in the mobile sector. There are many different brands, run-time platforms, networks and many other factors all contributing to the fact that there is a large variety of mobile devices on the market and, as a consequence, a number of problems with device-specific software [7]. It could be argued that it is almost impossible to write a single version of a mobile application that could run on every mobile device available. In addition, challenges related to data presentation (type and content of data may vary: text, image, sound, video, content streaming, presence, etc.) and device-specific features (screen resolution/display size, keyboard, mouse, touch screen, motion control, performance differences, cross-platform incompatibility, etc.) limit the full invocation of mobile technology in information management. As can be seen, fragmentation is wide-ranging and diverse in the mobile sector, and there are many technology-related issues, including the maturity of mobile information technology.

This situation described above creates an interesting research frame for exploration of this theme. Next, Section III presents one approach (dynamic user interfaces) for solving issues related to the fragmentation problem in mobile technology.

III. Dynamic User Interfaces

In the telecommunications sector and especially in mobile technology the main focus is increasingly moving away from improving the performance of data transfer and devices, toward applications, usability, and service configurability. In addition to workers needing ubiquitous access to critical information of the organization and knowledge exchange [2], there is a growing desire for more agile, adaptive, real-time use and features in mobile devices. One potential solution is the *dynamic user interface* of an individualized application, which also allows a company-specific customized data screen if necessary. The role of the *dynamic user interface* is significant in a business environment where real-time and up-to-date data - and also real-time reporting - is essential [1],[3].

The ongoing project entitled *Adaptive Software Services* (Mukautuvat Ohjelmisto Palvelut - MOP in Finnish) was initiated at the beginning of 2011 by the Tampere University of Technology (TUT) [8] with a consortium consisting of three Finnish ICT companies and the City of Pori. The two-year project is funded mainly by the Finnish Funding Agency for Technology and Innovation (Tekes) [26] and also partly by the participating organizations. The general goal of the project is to research software and information technologies and also deploy new techniques that can be useful in solving the problems and challenges related to the fragmentation issues in mobile technology described above. In general, the aim is to study how user interfaces can be adapted according to different needs and contexts. The project focuses on technologies that can enable context-based adaptive user interfaces and also the

distribution options for an interface that can be adapted according to the situation. More specifically, the research concentrates on the possibility of using *adapter software* and a centralized management system to update UIs dynamically. The main idea is that the adapter can update its UIs and extend functionalities by retrieving newer specification files from a system managing the UIs and their distribution. The research target is cross-platform solutions that provide software components and composed services that can be adapted in run-time for the preferred context or situation. The ultimate goal of the framework studied is to allow users to create applications, distribute applications to desired clients, use applications on any device with compatible adapter software, and to gather data. This study, and its results, supports the research related to enabler technologies and their implementation in the mobile information technology context.

IV. Flexibly Managed User Interfaces for Mobile Applications

This section describes the overall system architecture of an information system for creating and distributing data gathering applications. An example of the application creation process is also given. Fig. 1 shows the main components of the system, which are the UI Application Service and Adapter Software. The *UI Application Service* is used by the *Developer* or *Power User*, who creates and maintains the applications available for the users. The *Adapter Software* uses the Representational state transfer (REST) Application Programming Interfaces (API) of the *UI Application Service* to retrieve, update, and submit data. Finally, the *User* uses the system with the *Adapter Software*, which is installed into his/her device.

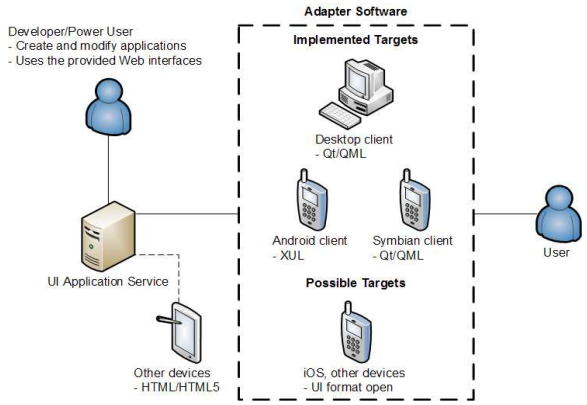


Fig. 1 Overview of the system

Currently there are three different platforms supported (*Implemented Targets* in Fig. 1): Android, Qt Symbian, and Qt Desktop. The Android client's UI format is Extensible User Interface Language (XUL) while both of the Qt versions use Qt Meta Language (QML) [27],[28],[29]. The *UI Application Service* is able to generate the correct UI files for each different type of *Adapter Software*. Also, each different *Adapter Software* is able to render the UI file according to the UI conventions of the platform/device. This allows the content of the applications distributed through the system to be similar on all devices but the look-and-feel and usage follows the UI concepts of the target device. It is also possible to support other devices, such as iOS, by implementing *Adapter Software* for the device. There have also been discussions about whether there should be a Hypertext Markup Language (HTML) [30] or HTML5 [31] compatible application interface. This would allow the usage of the system on an even greater range of devices.

The main goal of the information system is to provide an easy and rapid way to create and distribute query applications and data collection forms to the users. The typical workflow of using the system is started by the *Developer*, who creates an application. The new application requires a name and type. The name of the application is visible to the *User* when he or she browses through the application list in the *Adapter Software*. The application type determines how the application is rendered in the adapter software. Current variants are "Menu" and "Navigable." In menu applications the forms can be selected individually and in navigable applications the forms are displayed in a successive manner – like in wizard dialogs. The developer may also set the rights to access the application. The access restriction can be made by user group or simply on a user basis.

The next step is to create the forms of the application. An example of the form creation process is shown in Fig. 2. On the left side of the figure there is the *Developer view* or IDE view. The right side of the figure shows how the form would be rendered on the adapter software running on Android. The available components are shown on the left side of Fig. 2 and the preview of the current form on the right. There are now ten different components available but it is possible to include components such as Image or File for extending the usability of the system. The *Properties* section shows the editable fields of the selected component. The form has the name *XULTest* and it has three input fields named *Label1*, *Label2,* and *Label3*. The first two components are Textboxes and the last one is a Combobox. The components may also have a placeholder text set in the *Value* field.

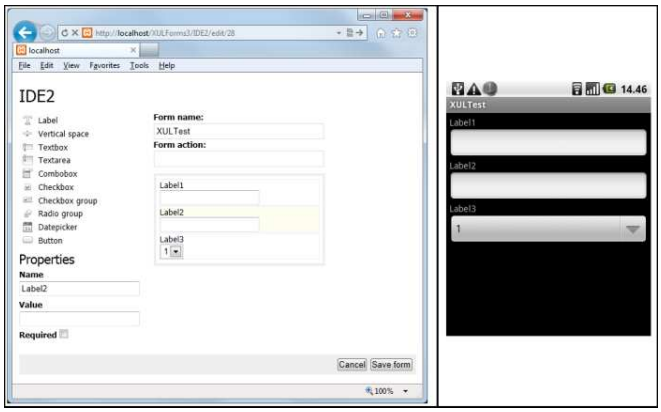


Fig. 2 Developer view IDE (left) and user interface on device (right)

The process continues by saving the form when it is finished and then creating as many new forms as are needed. When all the forms have been created, the users may start using the new application once their adapter software has retrieved the updated application definitions. The *UI Application Service* will also automatically create the database entries for data gathering and reporting purposes. The developer does not need to worry about how the data will be stored, as the UI Application Service will perform the task. The forms can also be edited if new components are required or corrections are needed to the old ones. The changes are propagated to the users once their adapter software updates itself.

The following Figures 3, 4, and 5 illustrate an example of the creation of an application, what the application looks like on an Android device and what kind of data have been gathered. Fig. 3 has similar components as seen in Fig. 2 but this time the form contains slightly different input fields. The left side of the figure is the IDE view and the right side is the corresponding view on the device. This application example consists of a total of three forms.

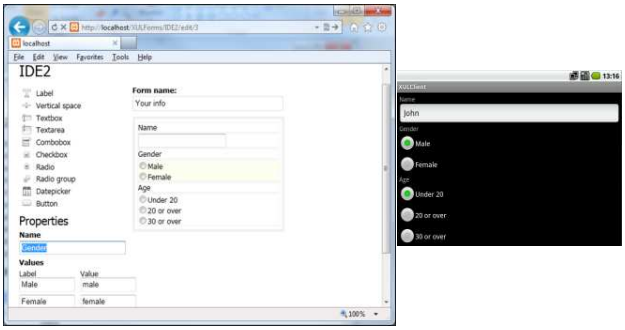


Fig. 3 Example of a query application (first form)

Fig. 4 (below) shows the other forms in the application. The corresponding IDE view is not shown for these forms but it follows the scheme seen in the earlier figures. It should be noted that the IDE or the adapter software does not do automatic paging but supports scrolling. The form seen in the IDE appears similarly in the device's adapter software, so it is up to the developer of the application to decide how to split the content. An example of this can be seen on the left side of Fig. 4, where the items seem to come to an abrupt stop, but in reality, the user can simply scroll down to see the rest of the form's content. The user may also navigate between the forms by pressing the *Back* and *Next* buttons. The *Next* button is replaced by the *Submit* button in the final form. After pressing *Submit*, the values given by the user are sent to the server.

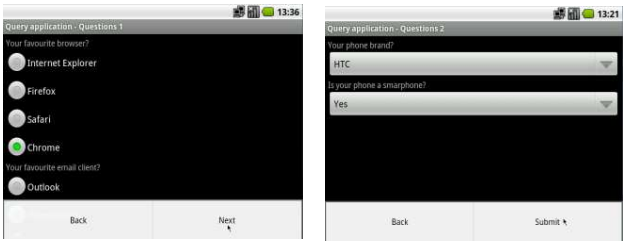


Fig. 4 Example of a query application (forms two and three)

All submissions of the application are listed in the IDE view with timestamp and user information. The full data of a particular submission can be seen by selecting it from the list. This is shown in Fig. 5 (below). The *Field* (name) column contains all the input fields which were created in the application's IDE view. The *Value* column contains the corresponding data value given by the user.

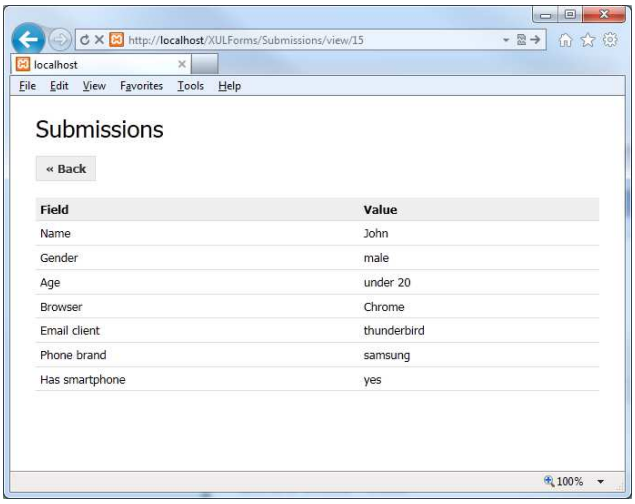


Fig. 5 Example of submission data gathered from the user

Figures 6 and 7 (below) show two screenshots of the adapter software running on Qt Simulator. The figures show what the adapter software would look like for the user - in this case test-user@example.org.

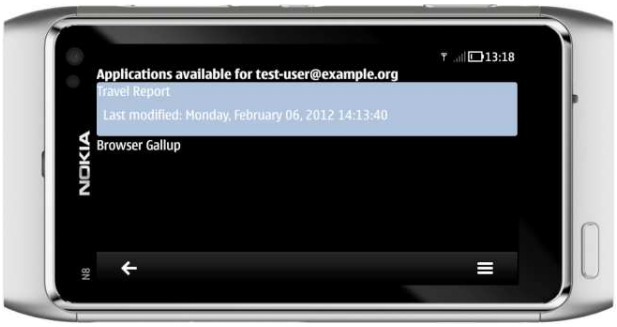


Fig. 6 Qt/QML-based adapter software running on Qt Simulator. Listing of the user's available applications

In this example the developer has created two applications: *Travel Report* and *Browser Gallup*. The Travel Report application is used as an example. Both of these applications are listed in Fig. 6. The *Travel Report* application is currently selected, and the adapter software shows detailed information about the application (in this case, the last modification date is shown). By selecting the application again, the adapter software will load the forms of the application into a new page as shown in Fig. 7.

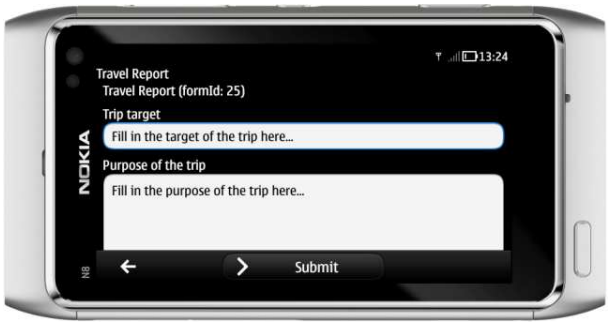


Fig. 7 Qt/QML-based adapter software running on Qt Simulator. Form view of the Application

Fig. 7 (above) shows the form of the *Travel Report* application. In this case, there is only one form but any number of forms may be created for the application. If there is more than one form the currently shown *Submit* button would be a *Next* button instead. Once the last form is reached, the button would be *Submit* again. When *Submit* is pressed, the adapter software gathers all the data from the input fields and sends the generated XML to the *UI Application Service*'s REST API.

V. Discussions

The adapter software approach seems to be a viable option when there is no need for more advanced components than those shown in the Fig. 2 IDE section. For example, when used for *query applications* (i.e. applications that are used to gather data from human users), the system has advantages as it is able to handle the application's creation, distribution, data collection, and data reporting. One other clear benefit of the system presented here is that this adapter type of model gives functionality to mobile network "offset" situations and guarantees that data are not lost from the mobile device. When the network connection is missing, then the Adapter application collects data to local storage for later usage. Of course, a web-based application might gather data on the server, but they have one problem in the network lost or "offline" situations. Old web-based applications do not handle "offline" situations at all and the data go missing. The new HTML5 standard gives some help in this, but when will it be supported by the browsers and how well? There is one further drawback in HTML5 local storage support, which is that the default amount of data is limited to 5 megabytes per application. Sometimes a mobile application database might be bigger, for example, when gathering some pictures.

Additionally, mobile sensors, for example, GPS, accelerometer, magnetometer, etc. could be easier to adapt to the application. Basically from the application developer or business point of view, we simply want to obtain e.g. location information from the user regarding where the data entry was made. In this model you simply put it in the text field of the GPS sensor included in your application and the adapter software handles the rest. For example, in a web application a special JavaScript API has to be used to put the location information in the text field. HTML5 and JavaScript have to be handled, but the web-based application does not guarantee the location information accuracy and the developer must have the knowledge to develop the web application. The adapter type of implementation, as here, does not require knowledge of HTML5 and JavaScript. One of the main concerns regarding the feasibility of the system is that the limited amount of available components restricts the developer from creating complex applications. However, there is a pilot program underway where these kinds of aspects will be tested and more detailed information on the feasibility of the system will come up during this pilot.

Therefore, the presented model helps shorten development cycles and organizations may develop a simple mobile application for targeted users very quickly. The approach studied and presented here can offer one solution for the increasing demands related to the agility, adaptability, and also individualistic use of mobile devices for information transfer. A short search has revealed that in the mobile application area there is a lack of *Adapter solutions* corresponding to those described here. However, one can assume that the technologies and solution presented here are indicative of potential emerging technologies in the mobile device industry in the imminent future.

VI. Summary

The paper deals with the strategic importance of mobile technology from the information management point of view. The paper discussed the opportunities and challenges that cutting-edge mobile technology can offer the information management field. The basis of the research was to try to find solutions for the challenges caused by the fragmentation typical of the mobile sector. In particular, the possibilities were studied of creating a framework for developing a flexible, device-independent and real-time updated user interface for mobile devices. The approach was based on the impression that flexible manageable user interfaces are becoming an important trend in mobile and ubiquity applications. The goal of the framework studied and presented here was to allow users to create applications, distribute applications for desired clients, use applications on any device with compatible adapter software, and to gather data. These kinds of information technologies and their features offer new opportunities for exploiting the usage of mobile devices in a more versatile way that at present. The proposed approach provides an innovative way to apply existing mobile technologies to enhance fluent

and suitable communication and information sharing in organizations. The technology presented here could be one of the key emerging technologies, through which the use of mobile devices could be expanded in the information management sector in the future.

REFERENCES

[1] K. Burden, "Business benefits of industry-specific mobile applications," Global Headquarters: 5 Speen Street Framingham (2005), MA 01701 USA P.508.872.8200 F.508.935.4015, Retrieved 7/9/2011 World Wide Web, http://www.idc.com

[2] G. B. Davis, "Anytime/anyplace computing and the future of the knowledge work," *Communications of the ACM*, vol. 45, no. 12, pp. 67-73, 2002.

[3] K. MacDonald, and J. Metzger, "Achieving tangible IT benefits in small physician practices," *California healthcare Foundation*, ihealth Reports, pp.1-36, Sep. 2002.

[4] A. Göker, and H. Myrhaug, "Evaluation of a mobile information system in context," *Information Processing & Management,* vol. 44, no. 1, pp. 39-65, 2008.

[5] S. L. Järvenpää, K. R. Lang, Y. Takeda, and V. K. Tuunainen, "Mobile commerce at crossroads," *Communications of the ACM*, vol. 46, no. 12, pp. 41-44, 2003.

[6] K. Siau, E. Lim, and Z. Shen, "Mobile commerce: Promises, challenges and research agenda," *Journal of Database Management,* vol. 12, no. 3, pp. 4-13, 2001.

[7] J. Van Gurp, A. Karhinen, and J. Bosch, "Mobile Service Oriented Architectures (MOSOA)," *Lecture Notes in Computer Science*, vol. 4025, pp. 1-15, 2006.

[8] Tampere University of Technology (TUT), Retrieved 15/3/12 World Wide Web, http://www.tut.fi/public

[9] C. Jou, "Automatic web table transcoding for mobile devices based on table classification," in *Proc. of IADIS Multi Conference on Computer Science and Information Systems,* pp. 143-150, Rome, July 2011.

[10] H. Jin, J. Y. Cho, and J. W. Park, "Dynamic user interface update using web based framework," in *Proc. of International Conference on Consumer Electronics (ICCE),* pp. 10-14, Las Vegas, Jan. 2007.

[11] J. Ye, and J. Herbert, "Framework for Use Interface Adaption," *Lecture Notes in Computer Science*, vol. 3196, pp. 167-174, 2004.

[12] N. Rosenthal, "Hybrid Qt/WebKit applications: Pushing the Limits of Web Development [Video]," *Presented in MeeGo Conference 2010*, Dublin, Nov. 2010. Retrieved 14/12/2011 World Wide Web, http://conference2010.meego.com/session/hybrid-qtwebkit-applications-pushing-limits-web-development

[13] The WebKit Open Source Project (2011), Retrieved 14/12/2011 World Wide Web, http://webkit.org

[14] J. Bishop, and N. Horspool, "Cross-Platform development: Software that lasts," *IEEE Computer*, vol. 39, no. 10, pp. 26-35, 2006.

[15] N. Mitrovic, and E. Mena, "Adaptive user interface for mobile devices," Interactive Systems: Design, Specification, and Verification," *Lecture Notes in Computer Science*, vol. 2545, pp. 29-43, 2002.

[16] N. Mitrovic, J. A. Rovo, and E. Mena, "ADUS: Indirect generation of user interfaces on wireless devices," in *Proc. of 15th International Workshop on Database and Expert Systems Applications (DEXA)*, pp. 662- 666, Zaragoza, Aug.-Sep. 2004.

[17] V. T. Vasconcelos, S. J. Gay, A. Ravara, N. Gesbert, and A. Z. Caldeira, "Dynamic interfaces," *Presented in FOOL conference 2009*, Savannah, Georgia, Jan. 2009.

[18] D. Buhalis, "E-airlines: Strategic and tactical use of ICTs in the airline industry," *Information and Management,* vol. 41, no. 7, pp. 805-825, 2004.

[19] G. H. Griffiths, and P. N. Finlay, "IS-enabled sustainable competitive advantage in financial services, retailing and manufacturing," *The Journal of Strategic Information Systems*, vol. 13, no. 1, pp. 29-59, 2004.

[20] M. N. Melville, K. Kraemer, and V. Gurbaxani, "Review: Information technology and organizational performance: an integrative model of it business value," *Journal of MIS Quarterly*, vol. 28, no. 2, pp. 283-322, 2004.

[21] S. J. Barnes, "Enterprise mobility: concept and examples," *International Journal of Mobile Communications*, vol. 1, no. 4, pp. 341-359, 2003.

[22] A. Krause, D. Hartl, F. Theis, M. Stangl, K. Gerauer, and A. Mehlhorn, "Mobile decision support for transplantation patient data," *International Journal of Medical Informatics,* vol. 73, issue 5, pp. 461-464, 2004.

[23] CIO insight (2003), "When will mobility add strategic value?," Retrieved 14/12/2011 World Wide Web, http://www.cioinsight.com/article2/

[24] H. Sheng, F. Fui-Hoon Nah, and K. Siau, "Strategic implications of mobile technology: A case study using Value-Focused Thinking," *The Journal of Strategic Information Systems*, vol. 14, no. 3, pp. 269-290, 2005.

[25] L. Cosgrove, "Applied wireless: Making wireless work in the business," *CIO Focus,* pp. 1-39, 2005.

[26] Finnish Funding Agency for Technology and Innovation (Tekes), Retrieved 15/3/2012 World Wide Web, http://www.tekes.fi/eng

[27] Mozilla Developer Network (2012), XML User Interface Language, Retrieved 10/2/2012 World Wide Web, https://developer.mozilla.org/En/XUL

[28] Nokia Corporation (2012a), Qt Meta Language - Introduction to the QML Language, Retrieved 10/2/2012 World Wide Web, http://developer.qt.nokia.com/doc/qt-4.8/qdeclarativeintroduction.html

[29] Nokia Corporation (2012b) Qt - Cross-platform application and UI framework, Retrieved 10/2/2012 World Wide Web, http://qt.nokia.com

[30] World Wide Web Consortium (W3C), HTML 4.01 Specification, W3C Recommendation, Dec 24, 1999, Retrieved 10/2/2012 World Wide Web, http://www.w3.org/TR/html401/

[31] World Wide Web Consortium (W3C), 2011, HTML5 - A vocabulary and associated APIs for HTML and XHTML, W3C Working Draft, May 25, Retrieved 10/2/2012 World Wide Web, http://www.w3.org/TR/html5/

P2

PUBLICATION II

Prototype System for Improving Manually Collected Data Quality
J. Soini, P. Sillberg, and P. Rantanen

Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA) 2014, pp. 99–106

Prototype System for Improving Manually Collected Data Quality

JARI SOINI, PEKKA SILLBERG and PETRI RANTANEN, Tampere University of Technology – Pori

Even nowadays, a great deal of measurement data is collected and also saved manually. In this kind of situation, there are phases when human error can easily occur and also when interpreting the typed collected measurement data could be difficult. This research aimed to discover resources for improving the quality of measurement data as well as better and more illustrative tracking of usage information in real time. The objective was both quality improvement of a specific measurement data collection process as well as the elimination of human error. This paper describes one reliable solution for this purpose, which improves the quality and also the visual presentation of manually collected data. The paper presents elements of the system developed for this aim and also the technology deployed along with its operational principles.

Categories and Subject Descriptors: H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; H.4.0 [**Information Systems Applications**] General

Additional Key Words and Phrases: Data quality, measurement process quality, data visualization, software applications

1. INTRODUCTION

The starting point of the research was to map out areas and activities of the public sector in which *savings* could be achieved by controlling, optimizing and intensifying operations. This research is a part of the ongoing two-year (2013-2014) Kiiaudata (Kiinteistöjärjestelmien datan älykäs analysointi – smart analysis of property systems data) project funded by Tekes [2014], where one of the main aims was to study potential new technologies for managing and controlling conditions in buildings in a smart way. In collaboration with the City of Pori, a survey was made about the points where measurement data is collected and also how said data is utilized. As the result of this mapping, it was decided to focus on the upgrading of measurement data collection and the new swimming pool was chosen as the research subject, as it is the city's most expensive individual building in terms of energy consumption.

The idea was that the maintenance staff would continue checking the physical measuring devices to ensure their conditions, but the collected data would be recorded with the developed system in contrast to the fully manual record keeping used in the past (i.e. pen and paper). The measurements produce information that can be used, for example, in consumption and condition tracking. For instance, analyses of alteration in energy consumption can be made by means of inclusive measurement and usage tracking based on it. Electricity, heat and water are examples for different measured energy currents. In many cases, the aforementioned currents can be tracked and anomalous situations can be reported automatically using modern computer controlled systems, but there still remain situations where manual work is required, especially when dealing with legacy systems.

Author's address: J. Soini, Department of Software Engineering, Tampere University of Technology – Pori, P.O.Box 300, FIN-28101 Pori, Finland; email: jari.o.soini@tut.fi.

 In: Z. Budimac (ed.): Proceedings of the 3nd Workshop of Software Quality, Analysis, Monitoring, Improvement, and Application (SQAMIA), Lovran, Croatia, 19.-22.9.2014, published at http://ceur-ws.org

There are several studies related to building automation systems and automatic sensor data collection, for example Cheng and Shen [2011] introduced wireless sensor networks based on embedded Linux. Nainwal et al. [2011] studied on remote surveillance and monitoring system utilizing wireless sensor networks, Vujović and Maksimović [2014] focused on utilizing Raspberry Pi as a building block of wireless sensor node, and Toshniwal and Conrad [2010] introduced a web-based sensor monitoring system on a Linux-based single board computer platform. However our focus was on systems where automatic sensors cannot be fully utilized. The work presented in this paper utilizes the findings of Soini et al. [2013], in which mobile devices, Global Positioning System (GPS) technology and route optimizations were combined in a real-time tracking service for delivery of goods.

The owners of the property chosen as the research subject – the new public indoor swimming pool of the City of Pori – were particularly interested in, for example, identifying development targets related to energy consumption measurement, development of the measurement process, early discovery of possible issues, and evaluation of the impacts of changes. For this research, a manually used digital data collection system has been developed as a collaboration project between Tampere University of Technology (TUT) and the City of Pori. The system developed facilitates the maintenance staff's work in registering and recording the measurement information as well as real-time tracking of usage information and perception of possible anomalous consumption situations.

2. PROBLEMS IN QUALITY OF MANUALLY COLLECTED DATA

Erroneous values are common when collecting and typing up data by hand, especially for long numeric values. Errors can also be very hard to detect, and it is difficult to know if the erroneous value was caused by an error with a meter or a correct value was simply mistyped by the person reading the meter. This was the problem observed and the starting point of this study. The assumption was that typing errors can be detected by software.

In some cases, it is not financially viable to replace measuring devices: many devices available today can be networked and contain automatic error detection or monitoring software, but this is not true for all devices, especially when taking into consideration many legacy devices. If these devices are seldom used or replacing them would be expensive, alternative approaches are required.

There are still many measuring devices that need to be checked periodically by a user. In practice this may require writing down the values by hand. In many places it is still common to use the basic pen-paper-and-Excel approach, in which the measurements are checked manually, written down and later inputted using datasheet software such as Microsoft Excel. The system presented here enables the pen-and-paper phase to be skipped. Using a management interface, reports of the values can be created and saved in various formats (such as .pdf or .xls). The paper describes simple client software, which uses Near Field Communication (NFC) [ISO 2013] tags to detect a measurement device called an "object" in the context of this paper. In the scope of this paper, an object means a monitored physical device (e.g. water meter).

3. SOLUTION – PROTOTYPE SYSTEM FOR COLLECTION OF CONSUMPTION DATA

The main idea behind the prototype system is to combine a typical web service, a mobile device with networking capability and a way to identify the object to implement a data gathering and reporting service. QR-Codes and Radio Frequency IDentification (RFID) [ISO 2008; Finkenzeller 2010] contactless proximity cards were the main candidates for identification purposes. RFID cards were chosen over QR-Codes as they should be more reliable to recognize in dimly lighted environments. It is also more convenient to touch the card instead of taking a photo of QR-Code when the space is limited.

Not all RFID cards, or tags, are alike as they vary on parameters such as operating frequency, data speed, distance of reading, power supply (passive, active, battery-assisted passive), and price. The

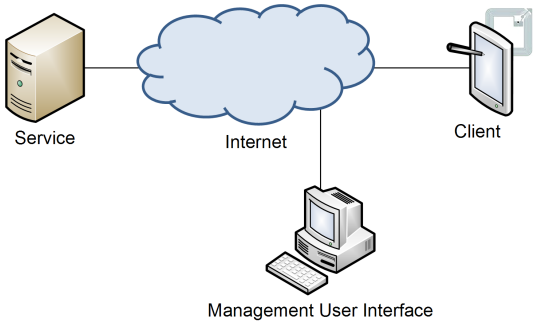


Fig. 1. System overview.

choice of parameters depends on the use case [Nummela 2010]. Typically, a low operating frequency correlates with low data speed and reading distance. An active power supply increases the price of the tag but enables the tag to operate without the support of a tag reader. We chose to use Near Field Communication (NFC) compatible tags as they are:

—relatively inexpensive

—they receive all the required power from the reader which reduces the need for maintenance

—the reading distance was not a crucial part of the system

—NFC capable smart phones and tablets are becoming more common.

For the purpose of this application, we are only interested in the unique ID which can be read from every tag. In our system this ID – i.e. a tag – is bound to an object. The user only has to touch the tag and the client software retrieves the correct data. Every object can be configured with various details:

—a common name (e.g. Water consumption)

—names of related gauges (e.g. Main water meter)

—the unit of the gauge (e.g. Cubic meters)

—warning limits for expected minimum and maximum daily increase (e.g. we expect that the gauge reading could increase by 50 to 100 units per day).

Figure 1 shows an overview of the system. The *Service* is available over the Internet where both *Management User Interface* and *Client* application can be connected. The service uses JavaScript Object Notation (JSON) to transmit data objects and it has two Representational state transfer (REST) interfaces, one for getting the gauge data and the other for posting the gauge data. It also supports user access control, but this feature is not currently used in the pilot phase of the system. The management user interface is a JavaScript-based web page accessible with a web browser. There the system administrator can configure a particular object and interpret the results sent by the client. For example, the results can be viewed as raw data or plotted as a chart. The *Client*, in Figure 1, is the main component that the end user is using. It is used to interact with tags, collect the data, and perform small scale on-site analysis of the data. The client application in our case is programmed for Android devices.

The prototype system is currently being tested in at the new swimming pool in City of Pori. There are three different gauges (water, electricity and central heating) which are being monitored. There are a couple members of the maintenance staff who operate the client device just to collect the gauge readings, and one person to oversee the changes in the collected data. All workers are operating the

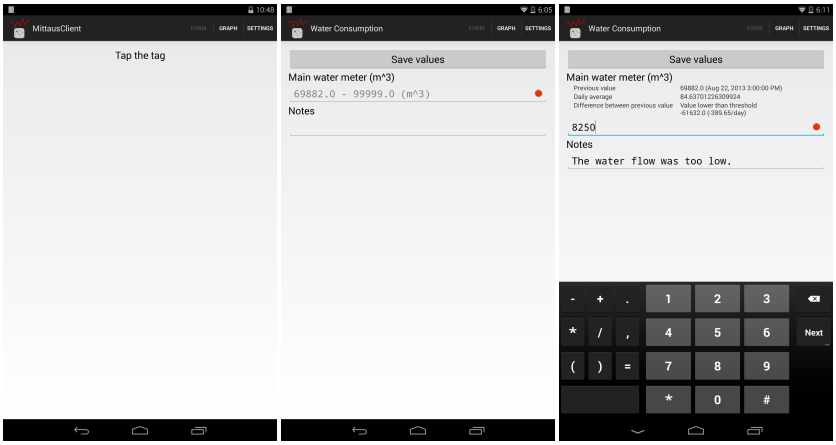


Fig. 2. Application screenshots, from left to right: initial view before a tag has been read, form view after the tag has been read, and finally, threshold value is below the defined limit.

client thru one shared device. So far the response from the staff has been enthusiastic about the data collection system, particularly of the ability to see the approximate costs of the facility immediately.

3.1 Information Collection

The client device and the NFC tag play an important role in collecting the meter readings. The information collection consists of three phases:

(1) identifying the object
(2) inputting the data
(3) saving the data.

Each of the three phases is explained in more detail in the following sub-sections.

3.1.1 *Identifying the Object.* The first step is to identify the object by touching the tag attached to the object. The tag will be automatically detected by the device. The tag detection is based on the unique ID found on every tag. In the current implementation these IDs can be mapped to objects using the service's management interface. This mapping is used by the client to detect which object is the current target and to show the correct object-dependent input fields. It could also be possible to extend the client software to enable mapping new tags for objects, which would make installing the overall system easier. This way the system installation could use bulk tags, which would be mapped to objects on the spot by the person performing the installation procedure. Whether mapping the tags on the device is required depends on the use case, and in our current scenario it was not a necessary feature mainly because of the relatively small amount of objects and tags. Also, as the main use of the client device is to gather information, it might be better to keep the software simpler to use by limiting the functionality available (see Figure 2).

The mapping information and the input field details can be synchronized with the service at any time, but in general, synchronization is performed only when specially requested. There are two reasons for this: firstly, the mapping and input field details change very rarely, making continuous synchronization a waste of network bandwidth; and secondly, in some cases the objects may be located in places with poor or non-existent network connectivity, making live synchronization difficult or even

impossible. The basic view before any tags have been detected is shown in Figure 2 (left), and the view after a tag has been selected is shown in Figure 2 (center). In the example case a very simple object is illustrated containing only two fields; a numerical input field for the *Main water meter*, which accepts values ranging from 69882 (the previous input value) to 99999, and a text input field for *Notes*.

3.1.2 *Inputting the Data.* Figure 2 shows the views of a detected object. The view in the center shows the basic view and the view on the right show the extended view. When the user taps any of the fields, additional information related to that specific field is shown: the previously given value with the timestamp of the input date, the daily average, and the difference of the currently typed value (if any) in relation to the previously given value. The purpose of the extended information is to give a quick glimpse of previous data, which can be used to detect possible errors in the readings and give the person using the device an idea of the possible values. In the example case (Figure 2, right), the red dot on the right hand side of the input field shows that a bad value has been given, and the user has typed a descriptive comment on the matter in the *Notes* section ("The water flow was too low"). Figure 3 (left side) shows the same case with the properly inputted value.

The value ranges used to detect and show warning situations are configured on the management web interface of the service. The ranges are numerical thresholds, which have either been calculated based on earlier data (e.g. it may be known how much water is used on average on a daily basis), or they may be based on physical limits (e.g. water consumption cannot be negative). The ranges can be simple minimum and maximum values, which should not be exceeded (e.g. voltage should stay between 10 and 15 volts) or cumulative limits (e.g. water consumption should not exceed ten cubic meters per day). The minimum and maximum values do not need previous values for accurate calculation of the warning threshold. In the case of cumulative limits, at least one single previous value is required. The previous values can be provided by the service when synchronizing the tag mappings and input fields or they can be results from previous use of the software. The warnings are meant to help the person typing the input values, and they are only "soft limits", i.e. they can be overridden if required. For example, it may be possible that a meter is giving an erroneous reading or for some reason much higher consumption is occurring. In this case it may be required to input a value that is outside the previously designated range. Inputting a value outside the range requires a confirmation from the user, and it will automatically be detected by the system and will pop up as an erroneous value on the management interface. It is also possible to generate an automatic notification, for example an email or SMS alert to be sent when an erroneous value is detected by the service, but in practice the notification will not be sent immediately if the data inputting process is performed in a location without network connectivity.

3.1.3 *Saving the Data.* After the user has inputted the desired data, the *Save values* button can be used to submit the results. The submit process may not necessarily start immediately. The values are stored locally on the device and can be viewed at any time, but when the actual result submission happens depends on the availability of the service. The client contains a background service which will periodically try to submit any unsent information. In our use case, the measuring devices themselves are located in an area of poor connectivity, but the users' workplace contains areas where the results can be submitted. The users generally carry the client device with them, thus allowing the automatic submission of the results when a network connection has been established. If instantaneous submission is required, other approaches should be considered such as providing wireless access by using a wireless router. The effects of periodic submit retries on battery life may vary. On one hand, turning the wireless radio off, and turning it on only when required may improve the battery life of the device. On the other hand, if the availability of the network connectivity is unknown, it may be difficult to establish the connection at timed intervals. In practice, many tablet and smart phone devices can

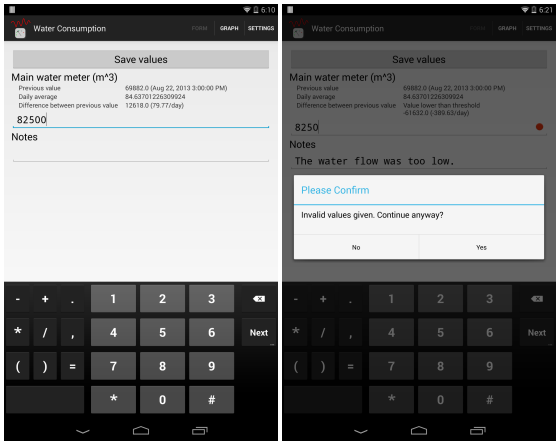


Fig. 3. Application screenshots: the left figure is the form view with data given by the user, the right figure asks for user confirmation before saving the data.

sustain a battery life of a whole day using the default power saving settings, thus only requiring the device to be charged when not needed, for example, outside working hours.

If the inputted values contain erroneous out-of-range values, a confirmation of values is required before the data can be saved and sent. The confirmation dialog is illustrated in Figure 3 (right).

3.2 Viewing the Results

The system allows the user to examine the collected data quickly on the client application and more thoroughly using the management user interface. Figure 4 illustrates the general idea of the different views:

—simple chart view of the client application on the left

—more complete analysis chart of the management user interface on the right.

The rationale for limiting client application features is to keep it as simple as possible and therefore to reduce the maintenance required for the application. It also helps to keep the device small enough for carrying around and for entering data. Also, the employee typing in the data might be more interested in seeing if the figures show any unexpected highs or lows, so he/she can react to the situation more quickly.

Both charts in Figure 4 contain the same data (consumption of water; x-axis time; y-axis consumption in cubic meters), but the view on the client application (Figure 4, left) is panned and zoomed in to show data between June 2013 and August 2013. The browser view (Figure 4, right) displays all of the data beginning from January 2012 and ending in May 2014. The upper chart shows the actual data and the lower chart illustrates the calculated daily average consumption. Between the charts there is a section with statistical information about the consumption. It shows the meter reading, date of the reading, and also approximated daily, weekly, and monthly costs in euros (by using a predefined average price per unit). The statistical information follows the pointer of the mouse so it is possible to see the same data from any point of the chart.

A surge in water consumption can be seen during July 2013 with consumption peaking at 400 cubic meters per day. This kind of information can be helpful for the maintenance team as it could be a sign of a leakage somewhere in the system. Fortunately, the peak was due to a scheduled maintenance of

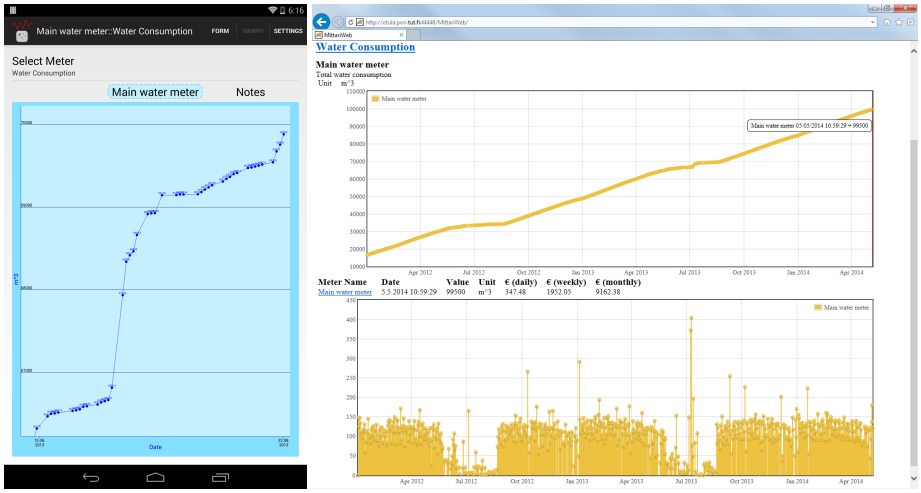


Fig. 4. Chart views as seen on mobile application (left) and on web browser (right).

the swimming pools. There are also many small consumption peaks and lows on the lower chart of the browser view. This occurred because the data was imported from the handwritten notes without exact time information. The collection time of the imported data is simply set at 12 noon, so it will cause fluctuation if the meter was actually read in the morning or evening. In the future as the data is collected directly to the system, the exact reading time can be stored, which will eliminate the fluctuation caused by unknown meter reading times.

The data shown in Figure 4 has been imported into the system from the actual water consumption data collected from the new public swimming pool located in the City of Pori. The facility has also been recording the consumption of central heating and the consumption of electrical energy. As the data comes from an actual facility, we had the opportunity to reflect on the consumption in terms of what had really happened. The data can be broken down into the following sections (see Figure 4, right side):

(1) January 2012 – June 2012, (winter & spring season, average consumption)
(2) June 2012 – August 2012 (summer maintenance, low consumption)
(3) August 2012 – June 2013 (fall, winter & spring season, average consumption)
(4) June 2013 – August 2013 (summer maintenance, from low to high consumption)
—Contains a surge of water consumption due to pools being emptied, overhauled, and then refilled.
(5) August 2013 – May 2014 (fall, winter & spring season, average consumption)

The data has been recorded by pen-and-paper, but is now being stored directly on the electronic database by using the system described in this paper. In fact, there are a lot of other digitally monitored and configurable parameters in the new swimming pool facility, but these three gauges (water consumption, central heating consumption and electrical energy consumption) are the only meters that still require old-fashioned manual reading.

4. DISCUSSION

The efficiency of the system greatly depends on the defined value ranges. If it is not possible to define clear ranges or the ranges remain vague, the possibility of error increases, and in this case the software works only as a pen-paper-and-Excel replacement. In practice, based on user feedback, the most

common source of error was grossly mistyped numbers, caused by lengthy numeric values (e.g. when writing down values it is easy to mix up 154763 and 157463, an error that can easily be detected by the software).

The software is more suitable for use cases where the meters are not read very often, but *do need* to be read manually periodically. If the meters need to be read continuously, for example several times a day, it may be more advisable to invest in meters with an automatic monitoring and warning system (if possible). On the other hand, if the meters are hardly ever checked, the basic pen-paper-and-Excel approach may be more feasible, and the resources required for setting up the system can be saved.

Then why not change the remaining analog meters? The comments from the facility's maintenance workers were that if they routinely read the meters every day, they can simultaneously monitor the condition of the nearby equipment and perform preventative maintenance if needed. Thus they can complete several tasks at once. It also helps to get a better grasp of the facility as a whole as they can see how much power or water is consumed daily.

5. SUMMARY

The paper presents a system for improving the quality of manually collected data. In many cases, especially in the public sector, there are many different points where manually measurement data collection is still practised. These situations usually relate to the monitoring of the operations of some physical devices, such as energy-related consumption measurement. The system introduced assists maintenance staff and also supports managers who are responsible for ensuring the correct operation of the devices. This system is one step towards more reliable and thus better quality measurement data, and it also improves the visual presentation of collected data for analysis. During the ongoing study, the system features will be extended and adapted for the purpose of monitoring patient rooms in the public sector health care environment.

REFERENCES

Xiaohui Cheng and Fanfan Shen. 2011. Design of the wireless sensor network communication terminal based on embedded Linux. *2011 IEEE 2nd International Conference on Software Engineering and Service Science* (July 2011), 598–601.

Klaus Finkenzeller. 2010. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication* (3rd ed.). Wiley.

ISO. 2008. *ISO/IEC 14443, Identification cards – Contactless integrated circuit cards – Proximity cards*.

ISO. 2013. *ISO/IEC 18092:2013, Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)*.

V Nainwal, P J Pramod, and S V Srikanth. 2011. Design and implementation of a remote surveillance and monitoring system using Wireless Sensor Networks. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, Vol. 5. 186–189.

Jussi Nummela. 2010. *Studies towards Utilizing Passive UHF RFID Technology in Paper Reel Supply Chains*. Doctoral dissertation. Tampere University of Technology.

Jari Soini, Timo Widbom, Jari Leppäniemi, Petri Rantanen, and Pekka Sillberg. 2013. Pilot system for transport confirmation with location awareness. In *Symposium GIS Ostrava 2013 - Geoinformatics for City Transformation*. Ostrava.

Tekes. 2014. Finnish Funding Agency for Technology and Innovation. (2014). http://www.tekes.fi/eng

Kailash Toshniwal and James M. Conrad. 2010. A web-based sensor monitoring system on a Linux-based single board computer platform. *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)* (March 2010), 371–374.

Vladimir Vujović and Mirjana Maksimović. 2014. Raspberry Pi as a Wireless Sensor Node : Performances and Constraints. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. Opatija, 1247–1252.

P3

PUBLICATION III

Portable Sensor System for Reliable Condition Measurement
J. Soini, P. Sillberg, P. Rantanen, and J. Nummela

Proceedings of the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) 2016, pp. 1397–1402
DOI: 10.1109/MIPRO.2016.7522320

Portable Sensor System for Reliable Condition Measurement

J. Soini *, P. Sillberg *, P. Rantanen * and J. Nummela **
* Tampere University of Technology/Pori Department, Pori, Finland
** Riffid Oy, Rauma, Finland
jari.o.soini@tut.fi

Abstract – Regarding sustainable development, there is a growing need to gather more and more various kinds of measurement, space, and consumption information about property. The necessity for property condition measurement is apparent and the appropriate circumstances, such as indoor air quality and suitable temperature, have an essential influence on comfort and welfare at work and, at the same time, have significance in terms of energy efficiency. This paper presents a portable prototype system for property condition measurement. The objective was to generate a reliable system that improves the quality and also the visual presentation of the collected data. The paper presents the components of the system and the technology utilized to implement the system. The results of piloting in a real-life environment, where particular focus was placed on both controlling energy efficiency and well-being at work, are also presented.

I. INTRODUCTION

Nowadays there is a growing interest in the indoor air condition of buildings, the operability of structure-technical equipment and systems, as well as in energy efficiency. Likewise, concerning sustainable development, there is a growing need to receive more diverse measuring, status, and consumption information on properties. Indoor air quality and temperature have an essential influence on working and living conditions. With condition measurement, fulfillment of condition objectives can be better ensured and possible anomalies can be detected early on. Companies in charge of the construction and maintenance of properties (and an increasing amount of private households and housing companies as well) profit from condition measurement in their activity. Property condition measurement offers the following benefits:

- energy conservation
- real-time consumption and status control
- improvement of working conditions
- malfunction management
- reliable history information of property behavior
- verification of the functionality of property automation
- alert notification.

In future, the need to receive more diverse measuring and status/freeze frame data on properties will grow. Measurement produces data that can be used, for example, to monitor consumption and circumstances for predictive maintenance purposes, or for the purposes of property automation. It is crucial to ensure that measurement systems work continuously and flawlessly, or if problems arise, the problem areas can be easily identified so that the necessary adjustments and possible corrective actions can be carried out. This paper presents one solution for this purpose - a portable prototype system for reliable condition measurement. The system developed facilitates the maintenance staff's work in detecting possible anomalous condition situations in properties.

Related research in this area has been conducted, for example, by Leong et al. [1] who have studied a Near Field Communication (NFC) and Bluetooth bridge system for connecting Bluetooth-enabled mobile devices to NFC-enabled consumer services. Ihasalo [2] has presented a tool for the construction and evaluation of continuous building performance measurement. Banerjee et al. [3] concentrate on the security aspects of sensors that collect sensitive or private data. Saari et al. [4] studied on collecting data from a sensor packet which was connected by wire directly to a single board computer. This was made by using embedded Linux and BeagleBone Black hardware, which is similar credit card–sized single-board computer as Raspberry Pi. The focus of that research was to collect data and to serve it over the Internet. Toshniwal et al. [5] have introduced a network-based sensor monitoring system that is portable for various applications. Voinescu et al. [6] describe in their research a USB device, which can be used as a wireless network gateway between ZigBee sensor nodes. These are some examples of studies related to this theme.

The structure of the paper is as follows: Section 2 represents the basis and the starting point of the system development work as well as briefly introduces the sensor technology utilized. Section 3 describes the architecture of our prototype system, data collection process, and the user interface for the graphical representation of collected data. Finally, section 4 summarizes the paper.

II. BACKGROUND – CASE: PORI HEALTH CARE CENTER

This case study was a part of the KiiauData (Smart analysis of property systems data, 2013-2015) project

funded by TEKES [7], where one of the main aims was to study potential new technologies for managing and controlling circumstances in buildings in a smart way. This case study is one example of the technology pilots carried out during the research project. This technology pilot - a portable sensor system - was developed as a collaboration project between Tampere University Technology (TUT) and the City of Pori. For this research case, the 5-storey hospital building of the City of Pori main health center was selected, which had undergone an extensive renovation, completed in the fall of 2014. The owners, i.e., the City of Pori, were particularly interested in identifying whether the property automation updated at the same time was functioning properly.

The research objective set was to implement a sensor system that would enable the flexible collection of the most reliable measurement data possible on changes in the conditions of the premises (rooms) and present the results in the clearest and most illustrative form possible. The research began by mapping out all the potential technologies for identifying changes in room conditions. In this case, the research was restricted to monitoring temperature and humidity data. An additional starting point for the system was that it should be *transferable* from one point to another, and *scalable* (addition of sensors) at a later date. Also, there was plenty of discussion on an appropriate measurement *results presentation* tool and methods from the aspect of analyzing and interpreting the collected data.

At the beginning of the case study, the sensor technology to be utilized in the system was selected. As a result of the specification and design stage, it was decided to build the system using RFID/NFC sensors. Considering the technical aspect of living conditions monitoring sensor networks, the application can be divided into three parts: 1) measuring the data, 2) transferring the data from the sensors to the service and 3) processing the data in the service. For part 2), one emerging approach for transferring data from sensors is NFC [8] or RFID (Radio Frequency Identification) technology [9], which was also utilized in this study.

Using RFID or NFC to gather data from sensors enables users to utilize a simple existing method for wireless data transfer, and thus develop scalable and cost efficient applications and solutions. NFC complements several widely used consumer level wireless technologies, by utilizing the key elements in existing standards for contactless card technology (ISO/IEC 14443 A&B and JIS-X 6319-4) and also enabling devices to share information at a distance of less than 4 centimeters with a maximum communication speed of 424 kbps. [10]. Choosing NFC as a communication method also reduces the energy requirement on the sensor side. All the energy needed for data communication will be received from the NFC reader, so the sensor battery capacity can be better utilized for data collection. This makes more frequent sampling possible while still maintaining long operation periods and lower maintenance requirements.

III. SENSOR SYSTEM IMPLEMENTATION

This chapter describes the implementation of the sensor system. First, the architecture of the system is described and then the selected sensor type and the data collection process. Finally, we introduce the user interface for the graphical representation of the collected data.

A. System Overview

Fig. 1 shows an overview of the system. The *Service* is available over the Internet where both the *Web User Interface* and *Data Collection* application can be connected. The service uses JavaScript Object Notation (JSON) to transmit data objects and has two REpresentational State Transfer (REST) interfaces, one for receiving the gauge data and the other for posting the data. The web user interface is a JavaScript-based web page accessible with a web browser. Here, the NFC tag collects the data independently, and maintenance staff is able to periodically transfer the collected data from the tag.

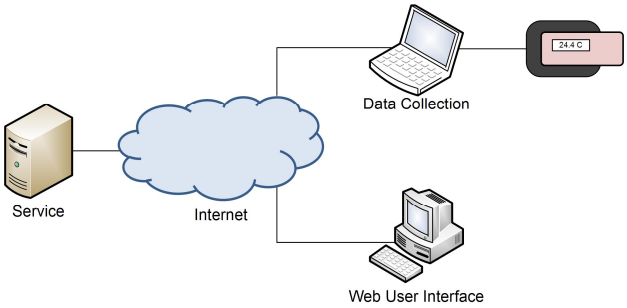


Figure 1. System overview.

Using the web interface, the collected data can be viewed embedded on a location map, plotted on a chart, or downloaded as raw data (as a comma-separated value file, CSV) for use with, for example, a spreadsheet application such as Microsoft Excel. A separate web page exists for system administrators for inserting new tag locations and uploading new data collected from the tags. In a practical use case, the tag locations are very rarely changed, and the web interface is generally used only for observing the tag data. Uploading new data is a relatively simple operation, in which the administrator uses the web browser to upload the log file transferred from the tag.

B. Sensors and Data Collection

The study began with a survey of the sensor types suitable for the purpose. As a result, 14 sensor types were found that could be adapted for the planned application. The key selection criterion was the customization capability of the sensors (e.g. code), as this feature ruled out most of the sensors found. The NFC-capable sensor KT-255F [11], which can store up to 16000 data points in its non-volatile memory, was chosen for this project. The device is shown on the left of Fig. 2 (below). It would also have been possible to buy the sensors separately and use a microcontroller or a single-board computer (for example, BeagleBone Black [12] or Raspberry Pi [13]) to implement an open source sensor device. In our case, temperature and humidity measurements were required and thus, a simple commercial device was a simpler and

easier solution. A proprietary device brings certain disadvantages when compared to a more open platform, such as dependency on the manufacturer's software and support and possible difficulties on moving from one manufacturer to another in the future, or making modifications to the existing hardware. The KT-255F sensor uses the FeliCa standard [14] in combination with a proprietary encrypted data format, which makes it difficult to read the tag using other than manufacturer certified reader software. Also, the encryption is not signed with a unique key, so the data can be read as long as you have the manufacturer's software and a FeliCa capable reader device, making encryption somewhat irrelevant. The data cannot be modified using an external reader, but the sensor can be reset to the default values. In practice, this means that any user with a suitable reader can read the data or reset the tag, but the user cannot insert malicious data. In our use case, this is not a major concern, as the tags are located in a public place (a hospital). The collected data, even though it is not published, is public, and in theory, you could gather the same data by simply walking around the building with a temperature meter in your hand. In addition, the data is used only for illustrating an overview of the conditions inside public areas, and accessing critical areas of the building requires privileged access, making tampering with the tags very difficult.

In our use case, however, the advantages outweigh any possible disadvantages. The tag device is in many areas situated in a publicly visible location, making a finalized product a more appropriate solution. The chosen commercial device is of sturdier build and is not as easily damaged if accidentally hit or dropped as a self-made sensor system would be. As an added benefit, the tag contains an LCD screen, which shows the current temperature, which according to the patients and the hospital staff was a nice feature, although not the most crucial part of the system.

Reading the tags is performed using a laptop with an external NFC tag reader and reader software. The software is capable of generating a PDF document with histograms and data values. The PDF format is unnecessarily difficult to convert to a format transferable to a database, and all the same graphs and charts can be generated by any other software, such as the web interface described in this paper. For that reason, it is much easier to collect the raw data only, which contains a comma-separated list of the values and their timestamps, and an ID unique to each tag, which can be used to detect automatically which tag was read. The data is automatically parsed by the service, and timestamps are compared to ensure the coherence of the data, and more specifically, to remove any duplicates in case the log files contain previously uploaded results. After the upload, the values are also checked for validity against a preset range of acceptable values (e.g., the temperature should be between 21 and 24 degrees Celsius). If a value is outside the range, an alert notification is automatically generated and shown in the web interface.

To summarize, for the sake of simplicity, the software is used to read the data log file, which is then uploaded by the system administrator or service personnel to the service using the administration web page. After the upload, the data will be immediately visible on the web interface. No other action is required, and the operation takes one to five minutes per tag, depending on the amount of data. The battery life of a tag is about two years, but the memory of the tag can contain only a limited amount of measurements. In our case, the tag sensors measure the temperature and humidity every 15 minutes, which fills the available memory in about five months. In practice, the measurements would be read more often than every five months, preferably once per month or even every two weeks for relatively up-to-date data. It should be noted that in this case, the data is used only for statistical and analytical purposes, i.e., verification of the property automation functionality, and there is no requirement for real-time data collection.

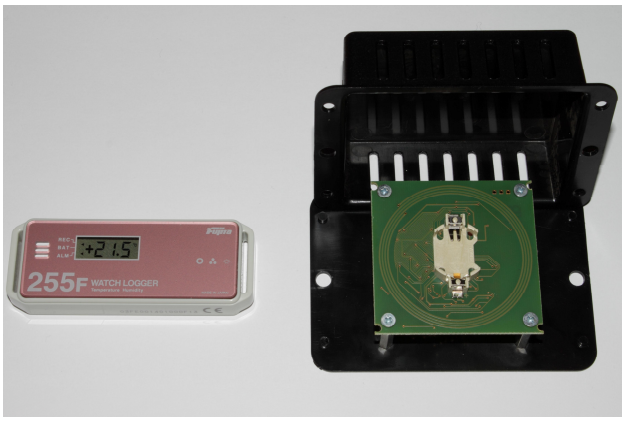


Figure 2. KT-255F Sensor and GoSense TH-Stat ID sensor.

In addition to the chosen sensor model KT-255F, other types were also sourced. Fig. 2 (right) presents another suitable option. However, this sensor, type name: GoSense TH-Stat ID NFC Temp & Humidity Sensor [15], varies clearly from the selected model, since it is not a ready-made commercial model nor is there any software available for configuring, controlling, and data handling. In some situations this can also be a benefit, since all data interfaces are open and specifications are freely available, which makes it possible to use this sensor in a wider area of applications. This sensor uses the standard ISO15693 air-interface for communication, which gives more flexibility in choosing the reader device compared to FeliCa, and can store up to 1 Mb measurement data. When considering this pilot study and its proof-of-concept type goals, using this sensor would have required considerably more effort in software development, which favored the use of the chosen KT-255F sensor.

When considering this application (living environment monitoring), it can naturally be carried out in different ways. One typical solution would be installing the necessary monitoring sensors permanently to the space under monitoring, and transfer measurement data through wires to the monitoring server. However, this approach requires more installation and cabling work than wireless options. The traditional way to utilize wireless data collection is to use radio links to transfer data, but then the battery life of the sensors may become an issue. The selected approach, using battery-powered data loggers and standardized NFC technology to deliver data from the sensors, makes it possible for sensors to gather a large

amount of data with high sample frequency and still keep the battery lifetime around two years, which is more than enough for this application. Along with this chosen architecture, the entire measuring system can be considered portable, as it can easily be taken to other locations for subsequent measuring periods.

C. *Web User Interface and Visualization*

The user interface is intended for the graphical representation of collected data. In addition to this, a meter location view was created to enable a quick overview of the collected data. This view is shown in Fig. 3. The web application supports internationalization (i18n) features but the data itself is not translated; it includes the names of the locations and the meters. The meter location view consists of the following sections:

- Top panel, which lists all the locations available for the authenticated user.
- Main view displaying locations and meters.
- Meter statistics, which can be seen when the cursor is moved on top of a meter marker. Fig. 4 illustrates the meter statistics window.
- Bottom toolbar with buttons *Show meters* and *Clear selections*.

The meter location view can show any kind of image and draw meters on top of it. The testing phase of the system consisted of installing sensors in the newly renovated facilities of the main health center of the City of Pori, mapping the sensor locations to floor plans, and finally defining acceptable boundaries for room temperature and relative humidity. In the event that the recorded data is outside the limits, an exclamation point will appear inside the corresponding meter marker. This tool can help the pinpointing of issues that occur in a certain room or floor.

Fig. 3 and Fig. 4 show floor plans of the main health center. The health center underwent extensive renovation work, and it was decided that collection of room temperature and humidity data was needed in order to verify a comfortable indoor environment. There are a total of 32 active sensors in this building, eight (8) for each floor, with a sampling period of 15 minutes.

Fig. 4 shows a close-up screenshot of a floor plan with an alert window opened. This alert appears in room number 24 on the third floor of the building (*3. Kerros Huone 24*). The window reports that both the humidity (8 % relative humidity) and the temperature have been too low (20.2 degrees Celsius), compared to the set limits. Other information displayed in the window is the average and standard deviation of all measured data types.

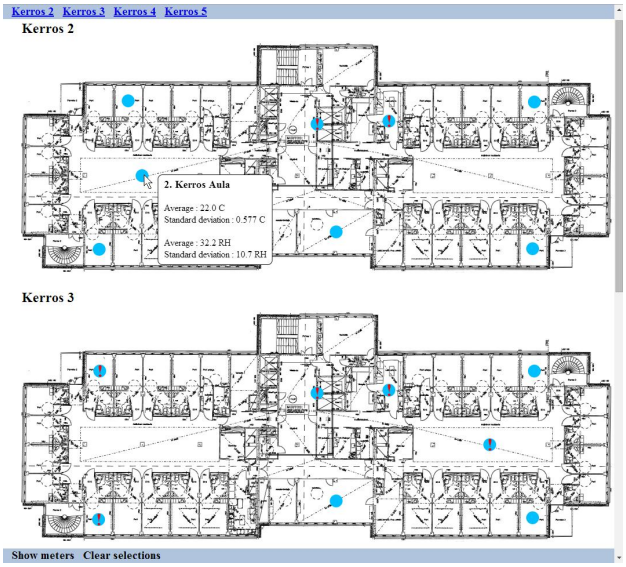


Figure 3. Meter locations overview.

The collected data can also be displayed in a chart view. The user may choose to show data from only one meter by double clicking the desired meter marker, or multiple meters can be shown by clicking any amount of meter markers, and then clicking *Show meters* (shown in Fig. 3). Fig. 5 (below) is a screenshot of a chart data view from the second floor lobby (*2. Kerros Aula*).

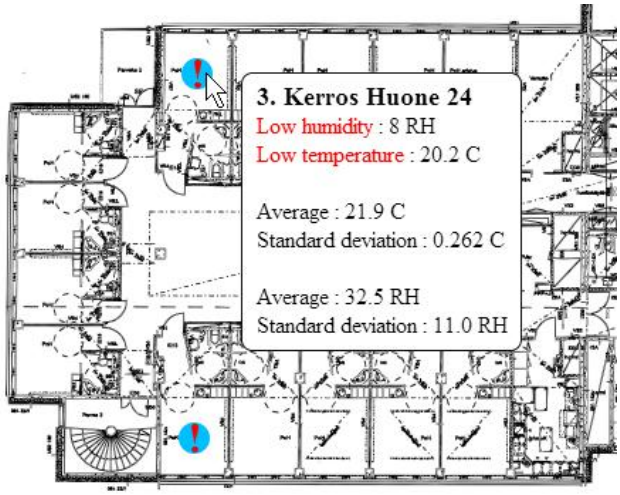


Figure 4. Close-up of floor plan with an alert window.

The upper chart zooms into the latest two weeks by default. The chart has options for zooming and panning for navigating through the data. The smaller chart on the bottom shows the overview of all collected data points. The chart can also be used for navigating through the data by selecting the desired range of dates.

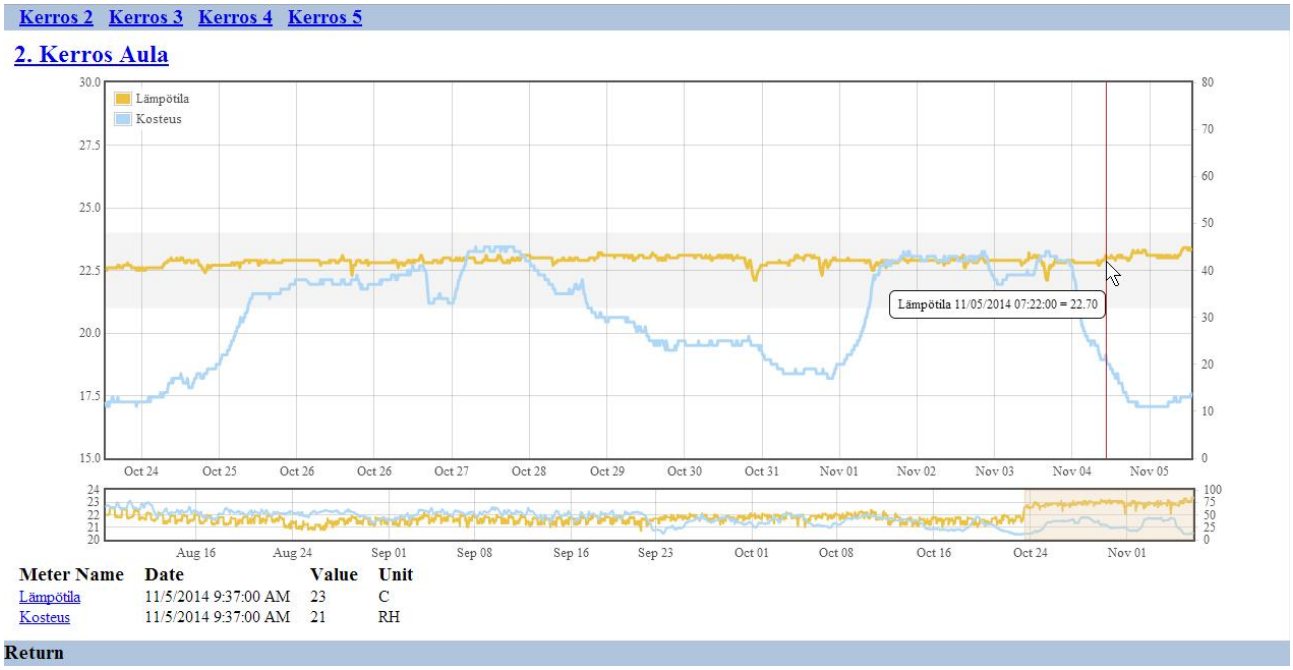


Figure 5. Meter value chart.

The example chart (Fig. 5) shows temperature (*Lämpötila*, higher line) in yellow and relative humidity (*Kosteus*, lower line) in blue. The chart can also draw the preferred zone for the temperature; here it has been defined as between 21 and 24 degrees Celsius. The cursor can be used to track the exact sensor value at a specific time and date shown in the tooltip. The *Return* button on the bottom toolbar can be used to go back to the list of locations view.

This prototype system has been in operation at the City of Pori main health care center since the fall of 2014. The primary goal of the system is to monitor the indoor conditions of the property and any changes that may occur over a long period of time. The health care center is under a large renovation project with an adjacent hospital building currently under construction. When the new construction project is complete, the sensor system will also be installed in the new location. The system can assist in detecting changes in the living and working conditions in both buildings during the renovation period, and after the projects are complete.

IV. Summary and Future Research

The paper presented a flexible and reliable prototype application for verifying changes in conditions inside a property and presents the measurement data collected to the system user in a clear manner. The portable condition measurement system makes it possible to reliably monitor and check the functionality of the property automation system at the renovated pilot target (main health center). The system introduced assists maintenance staff and also supports managers who are responsible for ensuring the correct operation of the devices. This system is one step toward more reliable measurement data, and it also improves the visual presentation of collected condition data for analysis. The portable, RFID-based, measurement sensor business is only just taking off globally. Conventional measurement sensors are still dominant, but in the near future both small- and low-current sensors will have a clear place on the market. In the future, the focus will be on studying how this technology can be applied to the packet transfer business. This technology does not need any wiring; sensors are easy to move and are easy to locate in terms of size and weight in different kinds of transfer packets. This is our next topic related to the sensor network research theme.

References

[1] C.Y. Leong, K.C. Ong, K.K. Tan, O.P. Gan, “Near Field Communication and Bluetooth Bridge System for Mobile Commerce”, IEEE International Conference on Mobile Commerce, Industrial Informatics, pp. 50-55. August 16-18, 2006.

[2] H. Ihasalo, “Transforming building automation data into building performance metrics - design, implementation and evaluation of use of a performance monitoring and management system”, Aalto University publication series Doctoral Dissertations, 26/2012, ISBN 978-952-60-4540-5 (electronic).

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

[4] 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), pp. 1037–1041, 2015.

[5] 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, pp. 371–374, March 2010.

[6] 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.

[7] Finnish Funding Agency for Technology and Innovation, Tekes, http://www.tekes.fi/en. (retrieved November 6, 2015)

[8] International Organization for Standardization. ISO/IEC 18092:2013, Information technology -- Telecommunications and information exchange between systems -- Near Field Communication -- Interface and Protocol (NFCIP-1).

http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56692 (retrieved November 6, 2015)

[9] Finkenzeller Klaus. RFID Handbook: Radio-frequency identification fundamentals and applications, Wiley, 1999.

[10] NFC Forum, What is NFC?, http://nfc-forum.org/what-is-nfc/ (retrieved November 6, 2015)

[11] Fujita Electric Works Ltd, KT-255F Specifications (Japanese), http://f-log.jp/?p=117 (retrieved November 6, 2015)

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

[13] 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.

[14] Sony Corporation, FeliCa Card User's Manual, http://www.sony.net/Products/felica/business/tech-support/data/card_usersmanual_2.0.pdf (retrieved November 6, 2015)

[15] GoSense Wireless Ltd, 20TH-StatID Datasheet, http://www.gosense-wireless.com/GoSense%20TH-StatID%20DS%20Rev1.pdf (retrieved November 6, 2015)

P4

PUBLICATION IV

Web-User-Interface System Utilizing rHMEI and Open Data for a Water Quality Analyzer

P. Sillberg, C. Veesommai, J. Soini, and H. Jaakkola

Information Modelling and Knowledge Bases XXIX 2018, pp. 420–428
DOI: 10.3233/978-1-61499-834-1-420

Web-User-Interface System Utilizing rHMEI and Open Data for a Water Quality Analyzer

Pekka SILLBERG [a,1], Chalisa VEESOMMAI [b], Jari SOINI [a] and Hannu JAAKKOLA [a]

[a] *Tampere University of Technology, Pori Department, Finland*
[b] *Keio University, Graduate School of Media and Governance, Japan*

Abstract. A clean environment is often taken for granted, but when a river or a lake becomes polluted, it might be hard for the general public to verify the condition. A simple visualization tool for checking the condition of water could help to inform the public and help to increase environmental awareness. With the emergence of Open Data and other openly available data sources, it is possible to create new and innovative applications. In this paper we present a web-based tool for reviewing the quality of water by applying the River Heavy Metal Evaluation Index (rHMEI) method together with open data on water quality in Finland.

Keywords. Visualization, Open Data, rHMEI, User Interface

1. Introduction

There is a saying that Finland is the land of a thousand lakes, although in fact there are 187 888 lakes larger than 500 square meters [1]. The value given to the protection of clean water is reflected by being a national concern in Finland [22], and continues to hold great interest for the Finnish public. When a web-based tool called Vesikartta[2] (literally water map in English) was published in 2016 for the public to review the condition of Finnish lakes and rives, it received 83 000 visits in one day [2]. In Finland, the current classification system emphasizes the ecological state of water areas, including several biological parameters such as fish populations, phytoplankton and aquatic vegetation. However, the quality of water is an essential component in the well-being and living conditions of biological organisms, thus monitoring of the water quality in lakes and rivers remains important. [3]

Water resource contamination by toxic substances is increasing, and is a cause of major concern to local people [4][5]. Several human activities may cause heavy metals to spread into ground water resources such as processing of metals, mining, discharge of agricultural waste, discharge of industrial waste, and use of pesticides containing

[1]Corresponding Author: Pekka Sillberg, Tampere University of Technology, Pori department, PO Box 300, FI-28101 Pori, Finland; E-mail: pekka.sillberg@tut.fi.

[2]http://paikkatieto.ymparisto.fi/vesikartta

compounds of heavy metals. [6][7] Concentrations of heavy metals exceeding a certain threshold level may affect the food chain and cause damage in aquatic organisms or to an industrial process. Safe levels of heavy metal concentrations in water are generally well established (e.g. [8][9][10]), but the synergistic effect of multiple heavy metals may affect the toxicity [11], and [9] remarks that the synergic effects of metals must be taken into consideration.

The River Heavy Metal Evaluation Index (rHMEI) is an environmental index that aggregates and rates the influence of each individual heavy metal parameter. It is a flexible tool for calculating and classifying water quality in a water resource in terms of heavy metals. The rHMEI can present the results of the environmental situation in terms of several heavy metal parameters and type of target category, such as irrigation or aquatic life. The index is also compared with six other methods. [12]

In this paper we present a web-based tool for reviewing the quality of water by applying the rHMEI method to an openly available data set. The algorithm to calculate the water-quality index is based on the findings of [12], and is the first implementation of rHMEI on an interactive web site. The open data set contains the physico-chemical determination values of water samples and is a repository of systematically collected data from various locations in Finland since the beginning of the 1960s.

The contents of this paper are as follows. Section 2 explains the background of the study; Section 3 describes the technical aspects of the implemented prototype application, explains the features of the user interface and presents a case study done with the prototype; finally, Section 4 concludes the paper and lists possible directions for future research.

2. Background

Visualization of data in a meaningful way is becoming more and more important as the amount of data increases. A good visualization also helps the user of the data to interpret it more easily. With visualization, more useful knowledge can be extracted from data. Visualizations of data allow the user to gain insight into the data and come up with new hypotheses and research questions about the phenomena behind the data. The analysis of Big Data, and especially Open Data from public administration may provide interesting and innovative solutions. Open resources such as Open Data can be seen as enablers for new applications [13] and it is expected to have at least the following advantages [14][15][16][17][18]: new jobs, new companies, new services and products, improved innovations, improved governance, more efficient use of the information, and even improved democracy. Concern about personal privacy related to openly available data has been raised [19], but environmental data—such as the data involved in this study—typically does not contain information that would breach anyone's privacy, so in that respect it would be safe to use this data.

Cyber-Physical Systems (CPSs) represent a new generation of digital systems where cyber entities and physical devices cooperate towards a set of common goals. [20] The CPS aims to be a complete solution, in order to integrate the dynamics of the changing environment in physical space, and dynamics among computers and networking in cyber space. [21] It is expected that the CPS will be used for the design and development of future environmental monitoring systems. In the use case of environmental monitoring,

the CPS can provide a solution for gathering, analyzing, and visualizing data from any device connected to the Internet. [22]

The prototype application is loosely based on the CPS concept by utilizing Open Data sources for data collection, the rHMEI [12] water quality evaluation method for analysis of data, and HTML5 web technologies for visualization. Our focus is on the visualization step which binds the results of physical and cyber spaces together. The result from physical space is the collected raw sensor data—in our case the Open Data provided by the Finnish Environment Institute—and the result from cyber space is the processed and evaluated environmental information. The result of the visualization step is an attempt to create an example solution based on Open Data in a way that can benefit the potential end users and the owner/publisher of the data. It is also an effort to disseminate the results of scientific research to non-specialist users.

How to increase public's awareness of environmental matters? It is suggested that the use of mobile technologies in education helps increasing students' awareness [23]. Furthermore, increasing environmental awareness with a computer application can be seen as a research problem in the field of human-computer interaction (HCI). While this topic is not assessed in this article, it could prove to be a fruitful method for studying whether the prototype application has effect on the user's environmental awareness or not. An approach of problem-solving in HCI research is given in [24] where they claim that it consists of three different problem types (empirical, conceptual, and constructive), and that the validity and reliability of the research can be evaluated by five criteria (significance, effectiveness, efficiency, transfer, and confidence).

3. Prototype Application

The prototype application for reviewing the quality of water is implemented as a HTML5 web page utilizing JavaScript for interactive features. The goal of the prototype is to provide an easy-to-access and a simple user interface with a minimal amount of configurable options. A simple classification (safe or toxic) for different categories should be clear enough for a user to determine the water quality. The application can be seen as an improvement over spreadsheets as it can speed up the analysis of the data by skipping an export and import phase completely.

3.1. Implementation

The web page follows single-page application design, and relies on client side processing for user interactions and external services for data retrieval. The data is loaded by using the asynchronous JavaScript and XML (AJAX) technique together with `jQuery` and `tabulator.js` libraries to produce interactive tables. The website requires the web browser to support at least ECMAScript 2015 (ES6). Although most of the code still uses the older JavaScript syntax, ES6 has a large number of useful improvements and new features [25] compared to the previous version.

The PIVET API[3]—the source of the Open Data utilized in the application—is based on Open Data Protocol (OData) version 3.0, which enables the use of RESTful calls to query data. From the API, we firstly needed to retrieve a list of sampling locations in a

[3]PIVET APIv2.0: http://rajapinnat.ymparisto.fi/api/vesla/2.0/

specified area, and secondly the actual data of the selected sampling location. In total, two different OData method calls are used by the application.

The list of sampling locations was acquired by calling the method `Paikka` on the OData API, and using Finnish municipalities[4] as a filter. A numerical identifier is given for each municipality, for example number 609 corresponds to Pori, and number 837 corresponds to Tampere. The identifier only changes if the municipality is disbanded, divided, or consolidated with another. The identifier is widely used in public administration data systems in Finland, and due to its static nature it is useful in all kinds of applications where municipalities need to be managed. Finally, the list of result entities is retrieved by calling the method `Result_Wide` and by using the desired location entity as a filtering parameter.

3.2. User Interface

Figure 1 shows a screenshot of the web page after completion of loading and rendering. The user may control the following aspects of the data to be retrieved: selection of municipality, selection of desired water area (i.e. sampling location, see Figure 2), selection of starting and ending date, and an option to filter out all values that are deemed safe (i.e. showing only the hazardous values, see Figure 4).

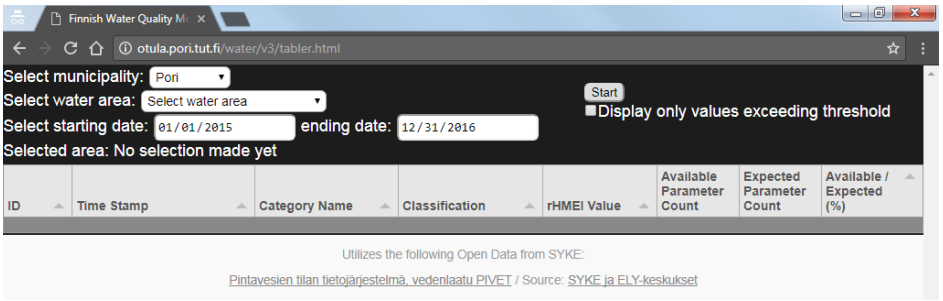


Figure 1. Screenshot of the initial view of the web page.

The table seen in Figure 1 contains the following columns: `ID` is the identifier of a sample used in the original dataset; `Time Stamp` shows the time when the sample was collected; `Category Name` explains what domain this information relates to; `Classification` indicates the meaning of the rHMEI value in plain language; `rHMEI Value` is the evaluated index value of the sample for the corresponding category; `Available Parameter Count` indicates the aggregate of suitable analytes found in the corresponding sample; `Expected Parameter Count` indicates the count of analytes that may be used for calculation of the corresponding category; `Available / Expected (%)` is simply the percentage figure of the two former columns.

To use the web page, the first step is to select one of the municipalities from the first drop-down menu. This menu simply contains a static list of active Finnish municipalities. By making this selection now, the user can filter out most of the unrelated water areas seen in the next step.

[4]List of Finnish municipalities in 2016: http://tilastokeskus.fi/meta/luokitukset/kunta/001-2016/ (in Finnish)

In the second step, after the user has selected the municipality, the list of water areas contained in the corresponding municipality is retrieved. In Figure 2, the selected municipality is Pori, and several sampling locations are shown in the selection list. The name of the water area might also be an acronym or contain more detailed information defined by the maintainer of the data, thus these names cannot be accurately translated. In this example, the water sampling location `Eteläjoki tie 272 mts` is selected. Each location is also associated with coordinate information, which enables the software to embed a map, or generate a corresponding link to any of the popular mapping web sites.

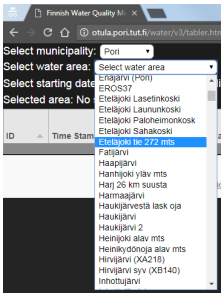


Figure 2. Screenshot of the list containing available water areas in the selected municipality.

The third and last step before retrieving and processing the data is to select the starting and ending date of the dataset. Some of the observation stations may contain data from the beginning of the 1960s, so it might be useful to limit the amount of data retrieved to the latest data only. This is just the default behavior of the web page, so users can retrieve the data from the beginning if they choose to do so. Once the user is ready, the data can be retrieved and processed by clicking the `Start` button.

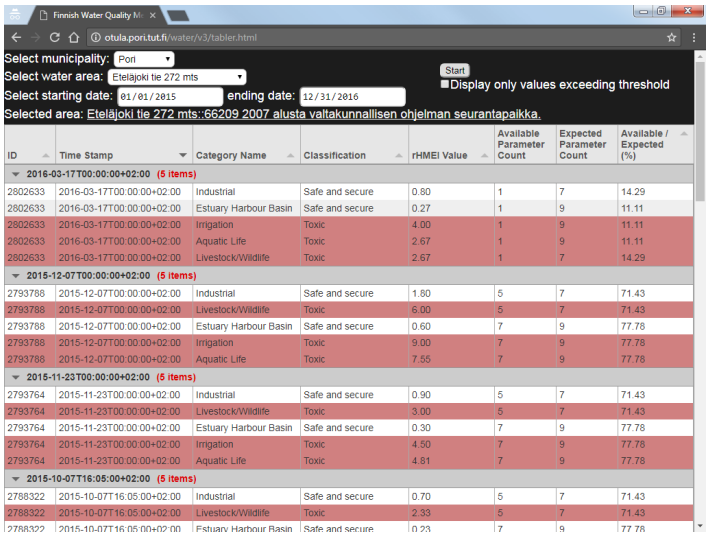


Figure 3. Screenshot of the unfiltered results table.

The `Selected area` field is updated with corresponding information after clicking the `Start` button. This field also includes a link to this location using Google Maps. The data is retrieved from the server, and is then processed for visualization. Figures 3 and 4 show two different views of the same data, unfiltered and filtered views, respectively. Filtering hides all values that are below the threshold level, thus it may help to spot occasional values deemed hazardous. The hazardous values may have a different background color (e.g. red) in order to differentiate them from the safe values.

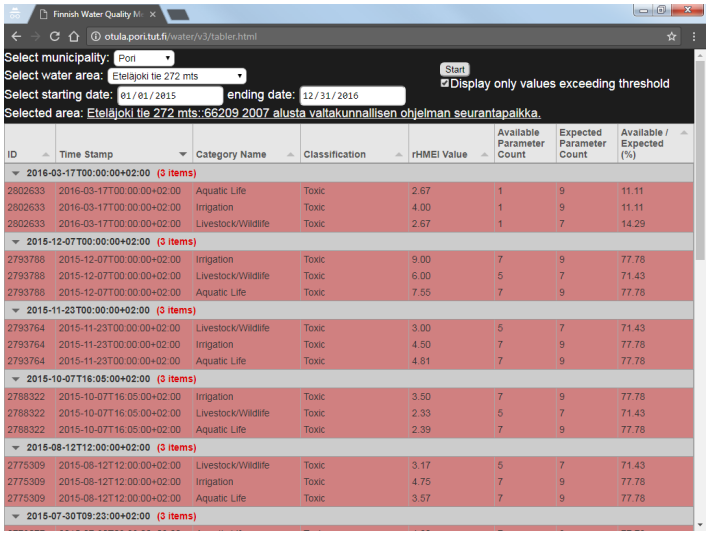


Figure 4. Screenshot of the filtered results table.

Each sample (a collection of water quality parameters) is swept in order to find the heavy metal analytes we are interested in. The values of these parameters are then processed against each pre-defined category. There are five different categories (`Aquatic Life`, `Estuary Harbour Basin`, `Industrial`, `Irrigation` and `Livestock/Wildlife`), and each of these contains a different set of heavy metal analytes and coefficients for producing the rHMEI value [12]. As the formula for calculating the rHMEI value is additive, the threshold value may be exceeded even if only one—sufficiently toxic—analyte is found from the sample data (for example, see rows with ID `2802633` in Figure 3). If more suitable analytes are available, the borderline cases may become more evident.

3.3. Case Study: Water Quality in Pori

We conducted an analysis of water resources in the municipality of Pori with the prototype application in order to find out what kind of information a regular user could be able to extract. The test was segmented into three time periods: 2010–2012 (3 years), 2013–2014 (2 years) and 2015–2016 (about $1\frac{1}{2}$ years as the database was last updated on the second quarter of the year 2016). There are 389 stations in the municipality of Pori and 84 of them had at least one measurement in the time frame of from 2010 to 2016. Table 1 summarizes the findings of each segment.

Table 1. Information from the municipality of Pori by means of the prototype application. Values indicate the number of stations.

Station Information	Segment I 2010–2012	Segment II 2013–2014	Segment III 2015–2016
Stations including data	70	67	48
Stations not including heavy metal analytes	14	14	10
Stations not exceeding any threshold*	21	27	19
Stations with an exceeding threshold*	35	26	19
Categories where the threshold is exceeded			
Aquatic Life	32	24	19
Irrigation	28	21	16
Livestock/Wildlife	25	18	16
Industrial	6	3	0
Estuary Harbour Basin	3	0	0

* excludes stations that did not include heavy metal analytes.

Approximately one-fifth (20–21 %) of the stations do not include heavy metal analytes and it has remained steady on all segments. In the Segment I, the percentage of completely safe stations (not including the stations without the measurements of heavy metal analytes) is 30 percent, while Segments II and III are at 40 percent. The remainder (Segments I, II and III; 50 %, 39 %, and 40 %, respectively) belongs to the stations that had at least one rHMEI value that was toxic. The amount of stations with an exceeding threshold per category appears to be at five to ten percentage point decline in each category. The reason for the decline cannot be interpreted from this data alone, so more knowledge of the surrounding environment will be needed.

The prototype application tells the number of the heavy metal analytes found, but it does not indicate which were found nor what is their significance to the calculated rHMEI value. The average count of available parameters was also recorded during the test. In the majority of the measurements, there was one out of nine ($\frac{1}{9}$) possible heavy metal analytes available (Segments I, II, and III; 70 %, 49 %, and 74 %, respectively). The second most common occurrence was seven out of nine ($\frac{7}{9}$) possible analytes, while the highest number of analytes was eight out of nine ($\frac{8}{9}$) which appeared in only one measurement.

4. Conclusion and Future Work

There is a lot of openly available data gathered by the public administration, at least in Finland, but the utilization of the data, such as visualizations and other applications, lags behind. Presenting the processed data in an intuitive and easily understandable way could make the service very popular and interesting for the public. In this paper we presented a tangible example of producing a simple user interface to review water quality by processing the water sample data collected in Finland. Currently, the prototype application is not targeted at any specific user group, and serves the needs of the authors as a verification tool. A user study should be carried out to test the usability and applicability of the prototype and to determine the users' interest in environmental matters.

As it was noted in Section 3.3, the application does not inform the user about which heavy metal analytes were used and what was their relative contribution to the calculation of rHMEI value. This should be included to provide more complete information. For further development ideas for the prototype application, the improvement of the look and feel and the addition of an interactive map of the corresponding measurement location were considered. For each location, the map could show an emoticon (e.g. happy, neutral, or sad) for each of the selected category (for example `Aquatic Life`) according to the analysis of the available data set. Extension of other fields, such as the quality of air, would complement the user's awareness of local environmental information. In addition, it would be interesting if it were possible to crowdsource the collection of water quality parameters. To implement this, it would be necessary to create a suitable back-end service for storing and validating the data. Improvements to the rHMEI method, categories, and coefficients can also be introduced into the application to make the provided information more accurate. It might also be useful to provide a summary of data (as seen on Table 1) to the user, as well as comparison between desired locations or between different time periods.

Acknowledgements

The research was made possible by the excellent and long-lasting co-operation between Tampere University of Technology, Pori Department (TUT Pori, Finland) and Keio University Shonan Fujisawa Campus (Keio SFC, Japan), and the jointly organized researcher exchange in the summer of 2016. The researcher exchange was supported partially by MEXT Grant-in-Aid for the Program for Leading Graduate School Graduate School of Media and Governance, and by the Global Environmental System Leaders (GESL) program, Keio University.

References

[1] H. Haarmann, *Modern Finland: Portrait of a Flourishing Society*. McFarland, Incorporated Publishers, 2016.

[2] Yle, "Vesien kuntokartan suosio räjähti - jopa 70 000 käyntiä tunnissa (in Finnish) [The popularity of the condition map of water areas skyrocketed - up to 70 000 visits per hour (title translated into English)]." http://yle.fi/uutiset/3-8878981, May 2016. Accessed December 23, 2016.

[3] The Water Protection Association of the River Kokemäenjoki, "Järvet ja joet (in Finnish) [Lakes and Rivers (title translated to English)]." http://kvvy.fi/?p=2495, November 2015. Accessed January 31, 2017.

[4] J. Duruibe, M. Ogwuegbu, and J. Egwurugwu, "Heavy metal pollution and human biotoxic effects," *International Journal of Physical Sciences*, vol. 2, no. 5, pp. 112–118, 2007.

[5] F. Xu, Z. Liu, Y. Cao, L. Qiu, J. Feng, F. Xu, and X. Tian, "Assessment of heavy metal contamination in urban river sediments in the Jiaozhou Bay catchment, Qingdao, China," *CATENA*, vol. 150, pp. 9–16, 2017.

[6] M. Hosseinpour, G. Lashkaripour, and P. Dehghan, "Assessing the effect of heavy metal concentrations (Fe, Pb, Zn, Ni, Cd, As, Cu, Cr) on the quality of adjacent groundwater resources of Khorasan steel complex," *International Journal of Plant, Animal and Environmental Sciences*, vol. 4, 2014.

[7] M. Sankhla, M. Kumari, M. Nandan, R. Kumar, and P. Agrawal, "Heavy metals contamination in water and their hazardous effect on human health-a review," *International Journal of Current Microbiology and Applied Sciences*, vol. 5, no. 10, pp. 759–766, 2016.

[8] World Health Organization, "Guidelines for drinking-water quality, fourth edition," Standard, World Health Organization, 2011.

[9] Environmental Protection Agency, "Parameters of Water Quality – Interpretation and Standards," Standard, Environmental Protection Agency, 2001.

[10] P. Tchounwou, C. Yedjou, A. Patlolla, and D. Sutton, "Heavy metal toxicity and the environment," in *Molecular, clinical and environmental toxicology*, pp. 133–164, Springer, 2012.

[11] C. Jayasumana, S. Gunatilake, and S. Siribaddana, "Simultaneous exposure to multiple heavy metals and glyphosate may contribute to Sri Lankan agricultural nephropathy," *BMC Nephrology*, vol. 16, no. 1, p. 103, 2015.

[12] C. Veesommai, Y. Kiyoki, P. Sillberg, J. Soini, H. Jaakkola, and P. Chawakitchareon, "The rSPA process realization: The creation of river heavy metal evaluation index (rHMEI) by using dimensional subspace of heavy metal," *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, vol. 7, no. 3, 2016.

[13] H. Jaakkola, T. Mäkinen, J. Henno, and J. Mäkelä, "Openn," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 608–615, May 2014.

[14] H. Jaakkola, T. Mäkinen, and A. Eteläaho, "Open data: Opportunities and challenges," in *Proceedings of the 15th International Conference on Computer Systems and Technologies*, CompSysTech '14, (New York, NY, USA), pp. 25–39, ACM, 2014.

[15] T. Turkki, "Nykyaikaa etsimässä – Suomen digitaalinen tulevaisuus," *EVA-raportti. Helsinki*, 2009.

[16] N. Huijboom and T. Van den Broek, "Open data: an international comparison of strategies," *European journal of ePractice*, vol. 12, no. 1, pp. 4–16, 2011.

[17] European Commission, "Open data – An engine for innovation, growth and transparent governance." http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:52011DC0882, December 2011. Accessed January 31, 2017.

[18] Cabinet Office of United Kingdom, "Open Data Charter." https://www.gov.uk/government/publications/open-data-charter, June 2013. Accessed January 31, 2017.

[19] H. Jaakkola, J. Henno, and J. Soini, *Data driven ecosystem - Perspectives and problems*, pp. 17–26. CEUR Workshop Proceedings, M. Jeusfeld c/o Redaktion Sun SITE, 2015.

[20] T. Sanislav, G. Mois, S. Folea, L. Miclea, G. Gambardella, and P. Prinetto, "A cloud-based cyber-physical system for environmental monitoring," in *2014 3rd Mediterranean Conference on Embedded Computing (MECO)*, pp. 6–9, IEEE, 2014.

[21] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 363–369, May 2008.

[22] G. Mois, T. Sanislav, and S. Folea, "A cyber-physical system for environmental monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 6, pp. 1463–1471, 2016.

[23] H. Uzunboylu, N. Cavus, and E. Ercag, "Using mobile learning to increase environmental awareness," *Computers & Education*, vol. 52, no. 2, pp. 381–389, 2009.

[24] A. Oulasvirta and K. Hornbæk, "HCI research as problem-solving," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, (New York, NY, USA), pp. 4956–4967, ACM, 2016.

[25] R. Engelschall, "ECMAScript 6 – New Features: Overview & Comparison." http://es6-features.org, 2016. Accessed April 21, 2017.

P5

PUBLICATION V

Toward Manageable Data Sources
P. Sillberg

Information Modelling and Knowledge Bases XXX 2019, pp. 101–111
DOI: 10.3233/978-1-61499-933-1-101

Toward Manageable Data Sources

Pekka SILLBERG [1]
Tampere University of Technology, Pori, Finland

Abstract. As information systems are producing vast amounts of data with ever increasing speed and diversity, the management of data is becoming an important part of gaining the information that we need. With this as the motivation, this paper proposes a Manageable Data Sources framework for the systematic management of data sources. The framework is derived from a new conceptual model of data processing: the Faucet-Sink-Drain model. The framework achieves two aims: the unification of data processing, and secondly, the componentization and decoupling of data processing related tasks. The framework is described and a reference architecture is laid out for the creation of a proof-of-concept implementation to solve the given use case.

Keywords. Framework, MDS, Conceptual model, Big Data, Data processing, Faucet-Sink-Drain model

1. Introduction

Information systems are producing data in ever increasing amounts, with greater and greater speed and diversity. At the same time, Internet of Things (IoT) devices are becoming smaller and more energy efficient, and require fewer resources to manufacture. These low power battery operated devices can stay connected to the network for up to ten years, or even longer [1]. This enables new kinds of applications, such as mounting sensor devices permanently into the structures of a building. As the quantity of devices grows larger, the more it increases the pressure on how the produced (big) data *can* and *should* be managed and utilized by the software. The ideal case is that it can be done in both a controllable and systematic way, while avoiding any error-prone ad-hoc implementations.

The use of patterns, frameworks, models, and re-usable code improves the quality of software. [2,3] By decomposing the system into smaller modules, the development of a piece of software becomes easier to modify and the re-usability of the code increases. For example, the Model-View-Controller (MVC) design pattern separates the application logic, data, and user interface into their own logical components. [4,5] To fully leverage the opportunities of big data in software engineering, we need to implement a decoupled and standardized component that is specialized in the handling of all data processing and searching-related requests.

Thus, the goal of this study is to identify a generic data processing model for creating a software framework to process data requests. The framework should make data

[1]Corresponding Author: Pekka Sillberg, Tampere University of Technology, Pori, P.O. Box 300, FI-28101 Pori, Finland; E-mail: pekka.sillberg@tut.fi.

processing easier and more organized, and improve the quality of the software on the data handling related operations. The framework must also be kept relatively simple. Simple and easily understandable concepts and terms may enable us to be more efficient when working with the big data challenges. Questions such as how to manage the ever increasing amount of data, how to utilize and visualize the data in a meaningful way, and how to programmatically extract knowledge and wisdom from the data are not necessarily answered in this study, but are nevertheless inspiring topics. The research question that this study attempts to solve can be formalized as follows:

> Does there exist a generic model or approach that enables the processing of all data in any given software application?

This paper proposes a new conceptual data processing model called Faucet-Sink-Drain model. The model is applied for a Manageable Data Sources (MDS) framework in order to create a systematic framework for the handling and processing of data. The framework covers the creation of data, includes the search, selection, and combination of real input data as well as any supplementary open data, and finishes with the outcome of the processed data (e.g., reports or visualizations). Furthermore, the framework gives the control of the data to the users by letting them choose and manage the desired data sources, data processors, and visualizations as best suits their need. The scope of this paper is to describe the model from the perspective of a single user and/or an instance, but there shall be no such limitation in the finalized framework.

The contents of this paper are as follows. Section 2 explains the background of the study; Section 3 concentrates on the specifications and definitions of the framework; finally, Section 4 concludes the paper and lists possible directions for future research.

2. Background

2.1. From Data to Wisdom

What is data and why do we need data, and where and how should we use it? Data is the raw building block of all information. This block, a symbol, represents the properties of objects and events [6]. Information is built upon data, knowledge is based on information, and wisdom requires knowledge [6,7]. Together they form a hierarchy, shown in Figure 1, oftentimes called the data-information-knowledge-wisdom hierarchy (DIKW). It is also known as the 'Knowledge Hierarchy,' 'Information Hierarchy,' or 'Knowledge Pyramid'. It is also referred to as the 'Wisdom Hierarchy' to promote the concept of wisdom over knowledge [7].

Literature has multiple definitions for each item in the DIKW hierarchy. In [7], the author reviewed 16 papers to find definitions and the essence of each of them. The author concluded that the difference between data and information is that plain data has no meaning as it is an unprocessed fact or observation. Information can be derived from the data by organizing it or by including a purpose or context, making it meaningful, useful, and relevant [7]. As a summary [7], knowledge can be seen as a combination of information, understanding, capability, experience, skills, and values. Knowledge can be differentiated into explicit and tacit knowledge, where the first is something that can be recorded programmatically but the latter is an embedded part of the human mind.

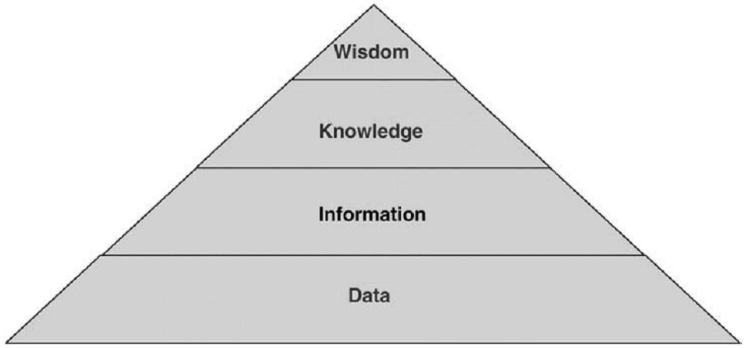


Figure 1. The hierarchy of data, information, knowledge, and wisdom (DIKW). [7]

2.2. Data Warehouse Design

The purpose of a Data Warehouse (DW) is to provide an architectural model to support the decision-making activities of an organization. It is achieved by extracting and storing variable data in a uniform format in a centralized database, which provides tools for querying, reporting, and information analysis. The definition by [8] says that "[a] data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions." The data produced by different sources can be imported to the DW by utilizing Extraction-Transformation-Loading (ETL) processes. There are two main design styles for setting up a data warehouse: bottom-up and top-down [9]. In the bottom-up model, smaller logical sections called Data Marts (DM) are solved first, and then integrated into the DW. The data included in a DM is specific to a certain business problem, and knowing the Business Model (BM) helps to analyze what data should be included in it. [9,10] In the top-down model, the whole dataset is fitted into the DW, and then the smaller DMs are designed from data provided by the DW [9].

Designing a complete DW comprises several phases such as the design of conceptual and logical schemas or ETL processes. Each phase in turn may have several competing techniques and patterns, but a lack of formal design guidelines for a whole DW system has been identified [10,11]. The framework introduced in this study resembles the concept of DW in several ways. Incidentally, MDS was designed independently without prior knowledge of the similarities with DW. The key difference is that with MDS, the input data can be imported with more flexibility and with less design effort. Data marts in the DW are more static and unchanging representations of an identified business problem, but in MDS, any stream of data (or combination) can be analogous to a data mart.

2.3. Big Data

Big data is certainly an interesting field for harnessing by the MDS framework, as all kinds of appliances and IoT devices produce data. Big data has been identified by [12] as consisting of three characterizing Vs: volume, velocity, and variety. The original definitional qualities of big data have since been varied in several other ways, typically by addition of more Vs (e.g. [13,14,15]). The other Vs introduced in subsequent sources included terms such as veracity, variability, visualization and value. Originally, the three

Vs were "intended to define proportional dimensions and challenges specific to big data" [16] in the same way as width, height, and depth are magnitudes of measures in real life three-dimensional space. The additional Vs can be used to understand various important topics when working with big data, but they are not aspects unique to big data only.

If the big data is too much to handle there exist solutions to reduce it to smaller, more manageable data chunks by the creation of coresets, subsets of the original data. [17,18]. Other means of analytics, as well as views on next generation DW research are studied in [19]. From the viewpoint of the MDS framework, big data is simply a lot of data, and the framework should be able to import and to process it based on criteria defined by the user. The framework's key point is the ability to handle different kinds of data sources in a systematic and controlled manner, with the goal of generating a suitable information for creating reports and the most value to the user.

3. The Framework: Manageable Data Sources

This section describes the design ideas behind the MDS framework. It will also clarify the conceptual model of data processing (i.e., the Faucet-Sink-Drain model) used in this paper. The framework aims to help the processing, combination, and management of data sources, by making them controllable and systematic. With the systematical approach, the traceability of operations (i.e., what steps were taken to reach the result) should become more evident and reproducible.

3.1. Overview

Let us start by introducing the key concepts of the model shown in Figure 2. There are five core components, the first (topmost in the figure) being a *faucet*, which appears to be open, and running a couple of *streams*. These streams go into a *sink*, and eventually into a *sieve* (i.e., an filter on the bottom of the sink). The sieve is connected to the last component, a *drain*. In this figure, the faucet simply appears from somewhere whereas the drain disappears. They are not necessarily connected to each other, but they are most likely connected to different systems which are not in the scope of this figure. An extended version of this concept will be explained together with the meaning of the added components in Figure 3. Term "stream" in the context of this paper does not relate to term streaming of media.

How does the imaginary version of this rather mundane appliance compare to its real world counterpart? Firstly, the faucet can be seen as the source of the data. The running water is a collection of data streams, and the sink is used to store and display the data. The sieve is a filter of the data with the capability of selecting and processing any chunk of any given data stream. The drain is a piping system to remove the excess amount of data in order to prevent information overflow in the sink. The drain may also be used for looping the processed data to another faucet and sink for refining the data even more. Secondly, this appliance has far greater adaptability than the physical one as it is easy to add or remove components.

It is not enough merely to collect all kinds of data sources together and display the data produced by them. When the data is combined together and processed in a certain way, the result will provide more data, which in turn can be processed again

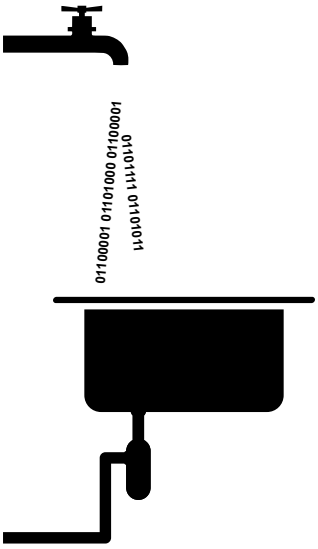


Figure 2. New conceptual model of data processing.

with some other data. If the data, selection criteria, and processing methods are chosen wisely, the data refinement cycle can eventually provide information that is valuable to its user. Therefore, the solution must be customizable and able to accommodate a feedback mechanism.

The framework strives to resolve this by being adaptable and able to grow seamlessly. Any number of data sources can be added, configurations for data visualizations and storage can be made based on stream criteria, and sieves can select and process any data stream. The extendability of the MDS framework is depicted in Figure 3, where an additional set of components (faucet, stream, sieve, and drain) have been "attached" to the sink to the sink (when compared to the starting point shown in Figure 2). From the software engineering and design point of view, the MDS framework also relies on the decoupling of data processing from the application logic in the same way as the MVC design pattern does with regard to model, view, and controller.

Let us examine Figure 3 more closely. The largest faucet located in the top left corner appears to be emitting two separate data streams. These two data streams[2] contain the following data pieces: `01100001 01101000 01100001` and `01101111 01101011`. In this case they are actually streams of binary data, usable by computers, but not so useful for humans. What could the user do to benefit from the generated data? The data can be refined in a series of loops, where each loop can improve the information value of the data. The feedback loop is the most essential part of this framework, because that is where all the data processing takes place in a systematic and controllable way.

A simple case of feedback processing is visualized on the right side of Figure 3. The added sieve is configured to select the "shorter" of the previous streams (`01101111 01101011`), process it, and forward the new processed data stream to a drain (piping system). If the drain is not connected to anything, the generated data will simply be dis-

[2]Streams actually have certain metadata attached to them, but they are omitted from the figure for the sake of simplicity.

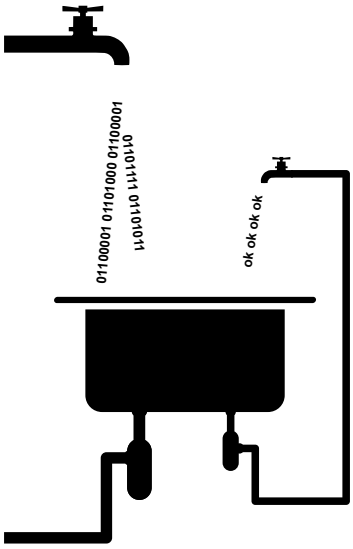


Figure 3. Extension of the conceptual model of data processing with a feedback mechanism.

carded. However, should we connect the drain to a new faucet, the data can then be directed to the data sink. The data stream that emerges from the faucet contains the following data: `ok ok ok ok`[3] which is an ASCII string representation of the previously selected binary data.

To recap, the MDS framework allows the addition of as many "feedback loops" as necessary. We can have any data stream (the old and the new) re-used, combined, and processed in additional sieves, which in turn create more data for additional faucets. If the chaining of the processed data is done wisely, the result will be useful information—and perhaps, knowledge.

3.2. Definitions, Glossary and Specifications

In this section, the framework's implementation-specific matters are dealt with. The key concepts and terms are defined as follows:

Data Component that forms the bulk of the content in the `DataSink`. It is basically of any type and format, and also includes the timestamp when the data was created. Each piece of data is associated to exactly one `DataStream` but the association is uni-directional, thus data is not aware of its owner.

DataStream Component which uniformly describes the type, format, and origin of the associated `Data`. Owns 0 to n `Data`, and is associated to exactly one `DataFaucet`. Not to be confused with with media streaming, or stream processing.

DataFaucet This component is the source of all content (i.e., data) and it also defines the physical characteristics of the source (e.g., location, update intervals and so on). The faucet has 0 to n `DataStreams`.

[3]The new stream in Figure 3 has four times more output compared to the input data; the reason for this is simply a more aesthetic look.

DataSieve Each `DataSieve` component has the ability to filter all data inside the sink, select the most interesting streams, and process and combine them into new streams. The processed output of the sieve—`DataStreams`—is transmitted to the attached drain. The sieve has exactly one `DataDrain`.

DataDrain The drain component is a transmission mechanism for the data. If left unconnected, the data will be discarded. Multiple faucets can be attached to a single drain, and all of these faucets will receive the same data. The drain may have 0 to n `DataFaucets`.

DataSink The central component where the data resides and can be observed. The sink will share the ownership of a `DataStream` once it has been emitted from the `DataFaucet`. Different kinds of `Visualizers` may be applied to any stream matching the defined `Criteria`. The sink may store and accumulate the data, or simply update the current data instance in the sink based on the `StreamOperator` defined on each stream. The `DataSink` knows 0 to n `DataFaucets`, and has 0 to n `DataStreams`. There must be at least one (1 to n) `DataSieve` to prevent "information overflow" in the sink.

DataProcessor A set of rules for processing `DataStreams` by the `DataSieve`. It defines any means necessary to create a new output data from the selected input data (e.g., addition, substraction, multiplication, logical AND & OR, set combination such as union & intersection, concatenations).

Criteria A set of rules for selecting the desired `DataStreams` that are available in the `DataSink`.

StreamOperator This controls how the `DataSink` needs to observe and handle the `Data` of a `DataStream` once the data has entered the sink. It says what needs to done to the data point to formulate a result, and how the result should be stored. For example, this could be a rule to accumulate the results for forming a time series of the data; to apply a differential and integral calculus to it; or to the continuous average and so on.

Type Definition of the (stream) type.

UUID Universally unique identifier.

Visualizer A visualizer may be used to create different views of the `DataStream` and the `Data` in the `DataSink`, such as documents, charts, maps, and reports. They are attachable and interchangeable, and do not modify the data.

Figure 4 depicts the reference architecture of the core components of the MDS framework. This high level view identifies the associations and quantitative relationships between these components. One important aspect is that this figure represents the case of a single user instance. This means that there may be other external components to which the components seen in Figure 4 are attached. For example, the `DataFaucet` may receive data from a drain created and defined by another user. Similarly, the `DataDrain` seen here can transmit its data to someone else's faucet.

Currently no implementations of the framework exist as the study is still in its early stages. Consequently, data formats regarding settings and description files have not been finalized, but the available techniques are constrained by the selected design criteria: compactness, splittability, exchangeability, and transferability. In practice, this means supporting at least the human-readable data serialization languages such as Extensible Markup Language (XML), JavaScript Object Notation (JSON), or YAML Ain't Markup

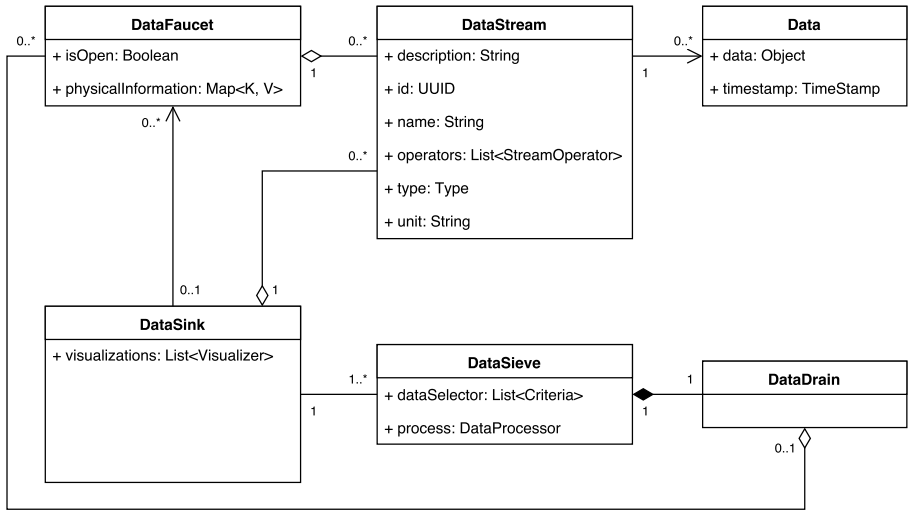


Figure 4. Reference architecture of Manageable Data Sources framework.

Language (YAML). Transferability, and easy sharing, would be important so that users could share their data sources with each other for more refined data, and information. This can be achieved by encoding the said files into barcodes or similar. If the files are splittable, it helps transferability, as individual file chunks can be fitted and shared with a storage medium such as barcodes or contactless integrated circuit cards (e.g., utilization of the NFC Data Exchange Format, NDEF). Being compact enough will allow the files to be fitted in a minimum amount of the previously mentioned media. The transmission of data does not have this kind of constraint, and it may use any means necessary to reach the system so that any internet connected device can be communicated with.

3.3. Discussion

The idea of the MDS framework has been introduced but there has not yet been any discussion of where it could be utilized. The use case scenarios described here are potential targets that might benefit from utilizing the MDS framework. Once the framework reaches the proof-of-concept maturity level, it should be benchmarked against at least one of the use cases.

Real-time Data The cars of today produce a lot of data and information with their electronic devices and sensors. A multitude of control units continuously monitor for the data in the Controller Area Network (CAN) bus. However, a fair share of that data cannot be accessed by the end user. Even if the on-board computer of the data can display information, it is typically limited to a pre-selected, non-customizable list of variables. The data is there, but the users have no means to access it without third party extensions and to make their own decision on what information to display. Ultimately, it is my car, and my data, and I want to have access to it.

So let us say that users could customize their access to their car's systems to select, process, and display any data produced by their car. OK, that would be nice, but where to next? There are multiple data sources in the car itself, as well as multiple other devices to

display data (e.g., dashboard, in-vehicle infotainment systems, laptops, mobile phones, and smart watches). In addition, there may be several users for the same car, but they may have different perspectives on the information that they need. An even more extended case is where the user has several cars—perhaps a fleet of cars—where they would like to have similar settings, preferences, and reporting features.

This is where the MDS framework comes in—by enabling the user to have their own systematically defined specification of what is important for them. The user's own pool of data, preferences, visualizations, reports and so on can follow them wherever they go, and allow them seamless integration with any available data source (i.e., by attaching their own `DataFaucet` to any openly available `DataDrain`). In an ideal case, the user would be able to (1) attach new data sources to their sink, (2) process and combine the incoming data, (3) refine the available data until the desired goal has been achieved, (4) control the results of the process (e.g., reports and visualizations), and (5) be able to release the results to other users.

Web Data Sources The data sources available around the Internet might not be providing a real-time data, but nevertheless, data processing and data source management could be a useful tool on monitoring of these sources. The following examples are chosen because I have had the opportunity to contribute in the development of the mentioned systems. In the first system [20], a third party Open Data source is being used to detect dangerously high heavy metal concentrations in water. This is a relatively simple use case to implement as the results of calculations are displayed in a tabulated format. Similarly, the second example [21] is quite a simple data collection and visualizer system, that has been used to gather the usage data of water, electricity and heat consumption in a public swimming pool in the City of Pori. In this case scatter and bar charts are used to visualize the consumption values. Both of the use cases could benefit from further data processing, such as the trend detection or implementation of any new analysis method that was not implemented in the original user interface. In order to achieve this, the data source as well all the chain of currently applied analysis methods should be transferable to another environment (for example, import from the application website to the user's personal computer). This transferable processing information, and the previously defined steps from 1 to 5, can be called a Faucet-Sink-Drain recipe.

Data Protection In the wake of General Data Protection Regulation (GDPR), the proposed framework could be even seen as a part of the underlying operating system where all data accessed by different applications require an authorization from the user first. The user may select which kind of data the application may read and process, and is that application allowed to make changes (e.g., add or remove) to the system-wide data or merely to the clone of the data that resides isolated in the application space. Basically, the operating system governs the biggest `DataSink` containing all the data, and each application have access to a "smaller" `DataSink` (i.e., the subset of the data). Applications can be seen as `DataFaucets` and/or `DataDrains` depending on their usage of the data. To be able to access the data on `DataSink`, the application must publish a list of the required and the optional data (i.e., `DataStreams` captured by `DataSieve`). The application developer would have to compose the list, and indicate if the application can not operate without the requested data. For example, a navigation application would probably be rather useless without location information, but the user might simply want to use that application because of the maps it can provide; the developer might accom-

modate to the user's wishes and offer a way to use the application without the navigation feature. Alternatively, the user might want to feed dummy location information to the application, but the application developer does not allow the use of application in this kind of situation.

4. Conclusion and Future Work

In this paper, I presented a new conceptual model of data processing and a framework called Manageable Data Sources for systematic management of different sources of data. The aim of the framework is to make data processing easier and more organized, and to improve the quality of the software on the data handling related operations.

The concept consists of five basic components: (1) faucets, (2) streams, (3) sink, (4) sieves, and (5) drains. The concept of MDS relies on the decoupling of the data processing component from the application logic in the same way as MVC design patterns with regard to model, view, and controller. Data processing in MDS is a continuous "feedback loop" of data, in which the data is refined bit by bit in order to provide more data and valuable information to the user. MDS is customizable to meet the requirements of different users. Finally, an example use case showing the potential for future research in the data source management field was introduced.

Future work will focus on the implementation of a proof-of-concept application which can be used to perform experiments and to collect evidence for evaluating the framework. The proof-of-concept implementation of the framework is to be evaluated by at least one of the designed methods, such as the Architecture Trade-off Analysis Method (ATAM) to find any possible defects and design flaws as early as possible [5,22]. Also, the applicability of the model in the example use cases requires more evidence and experiments.

References

[1] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[2] T. T. Nguyen, *Improving software quality with programming patterns*. PhD thesis, Iowa State University, 2013.

[3] J. Greenfield and K. Short, "Software factories: Assembling applications with patterns, models, frameworks and tools," in *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, (New York, NY, USA), pp. 16–27, ACM, 2003.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, A System of Patterns*. Wiley, 2013.

[5] K. Koskimies and T. Mikkonen, *Ohjelmistoarkkitehtuurit*. Talentum, 2005.

[6] R. L. Ackoff, "From Data to Wisdom," *Journal of Applied Systems Analysis*, vol. 16, no. 1, pp. 3–9, 1989.

[7] J. Rowley, "The wisdom hierarchy: representations of the DIKW hierarchy," *Journal of Information Science*, vol. 33, no. 2, pp. 163–180, 2007.

[8] W. H. Inmon, *Building the Data Warehouse*. New York, NY, USA: John Wiley & Sons, Inc., 1992.

[9] D. Bourgeois, *Information Systems for Business and Beyond*. The Saylor Foundation, 2014.

[10] S. Luján-Mora and J. Trujillo, "A Comprehensive Method for Data Warehouse Design," in *Proceedings of the 5th Intl. Workshop on Design and Management of Data Warehouses (DMDW'03)*, pp. 1–14, 2003.

[11] V. Peralta and R. Ruggia, “Using Design Guidelines to Improve Data Warehouse Logical Design,” in *DMDW*, 2003.

[12] D. Laney, “3-D Data Management: Controlling Data Volume, Velocity and Variety.” https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf, February 2001. Accessed January 18, 2018.

[13] M. Van Rijmenam, *Think Bigger: Developing a Successful Big Data Strategy for Your Business*. AMACOM, 2014.

[14] IBM, “Extracting Business Value from the 4 V’s of Big Data.” http://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data, January 2016. Accessed January 24, 2018.

[15] R. J. Self, “Impact of the "12 Vs" of Big Data on Questions of Ethics, Trust, Stewardship and Governance of Analytics.” http://computing.derby.ac.uk/c/wp-content/uploads/2014/11/Self_Richard_A2014.pdf, 2014. Accessed January 24, 2018.

[16] D. Laney, “Batman on Big Data.” https://blogs.gartner.com/doug-laney/batman-on-big-data, November 2013. Accessed January 18, 2018.

[17] D. Feldman, M. Schmidt, and C. Sohler, *Turning Big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering*, pp. 1434–1453. Society for Industrial and Applied Mathematics, 2013.

[18] D. Feldman, M. Volkov, and D. Rus, “Dimensionality Reduction of Massive Sparse Datasets Using Coresets,” in *Advances in Neural Information Processing Systems 29*, pp. 2766–2774, Curran Associates, Inc., 2016.

[19] A. Cuzzocrea, I.-Y. Song, and K. C. Davis, “Analytics over Large-scale Multidimensional Data: The Big Data Revolution!,” in *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP*, DOLAP ’11, (New York, NY, USA), pp. 101–104, ACM, 2011.

[20] P. Sillberg, C. Veesommai, J. Soini, and H. Jaakkola, *Web-User-Interface System Utilizing rHMEI and Open Data for a Water Quality Analyzer*, pp. 420–428. Frontiers in Artificial Intelligence and Applications, Netherlands: IOS Press, 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, September 19-22, 2014, Lovran, Croatia*, pp. 99–106, 2014.

[22] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. SEI series in software engineering, Addison-Wesley, 2002.

P6

PUBLICATION VI

Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data

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

Intelligent Systems: Theory, Research and Innovation in Applications. Ed. by R. Jardim-Goncalves, V. Sgurev, V. Jotsov, and J. Kacprzyk 2020, pp. 99–119
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.

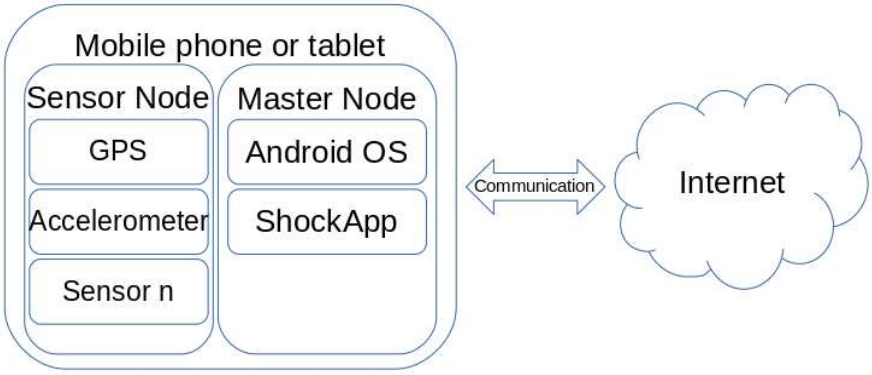


Fig. 1. The modified data gathering model.

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]

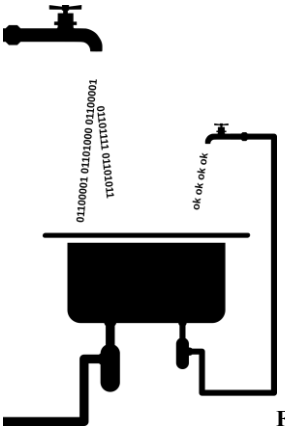


Fig. 2. Abstract data processing model. [10]

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.

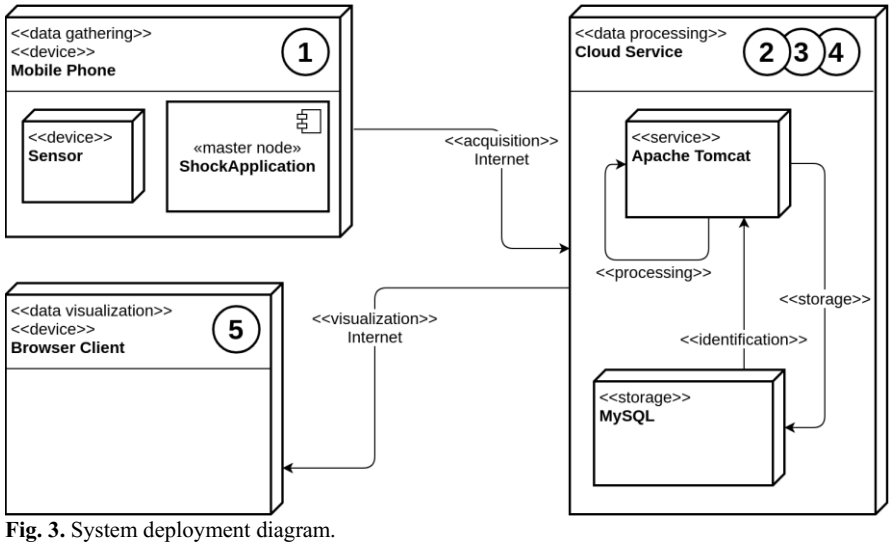


Fig. 3. System deployment diagram.

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.

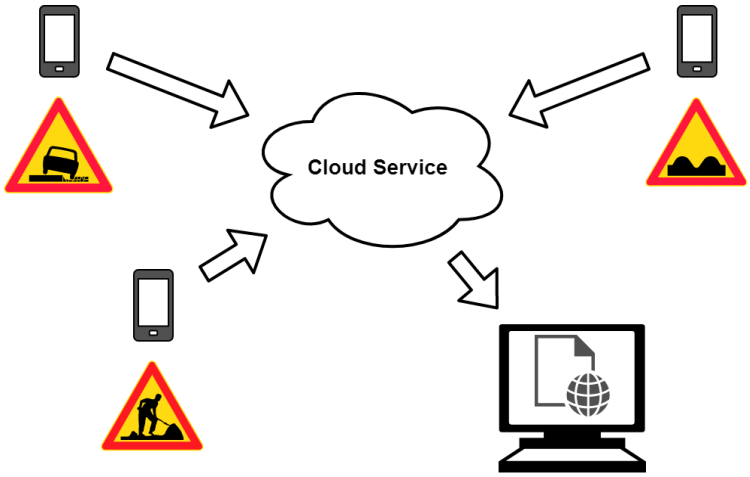


Fig. 4. High-level diagram of the test setup.

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.

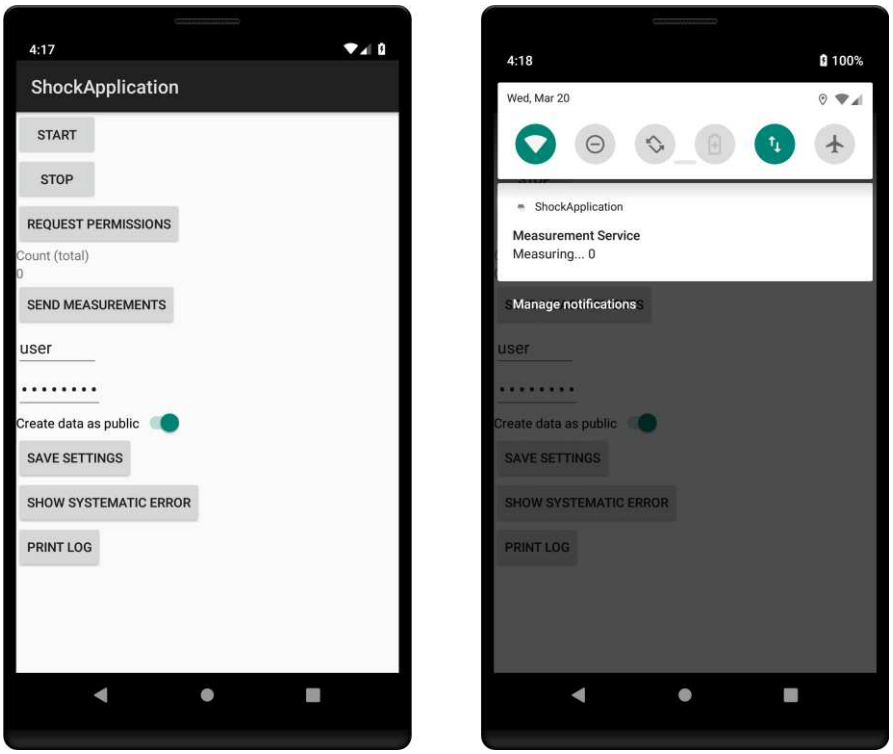


Fig. 5. The Android test client.

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.

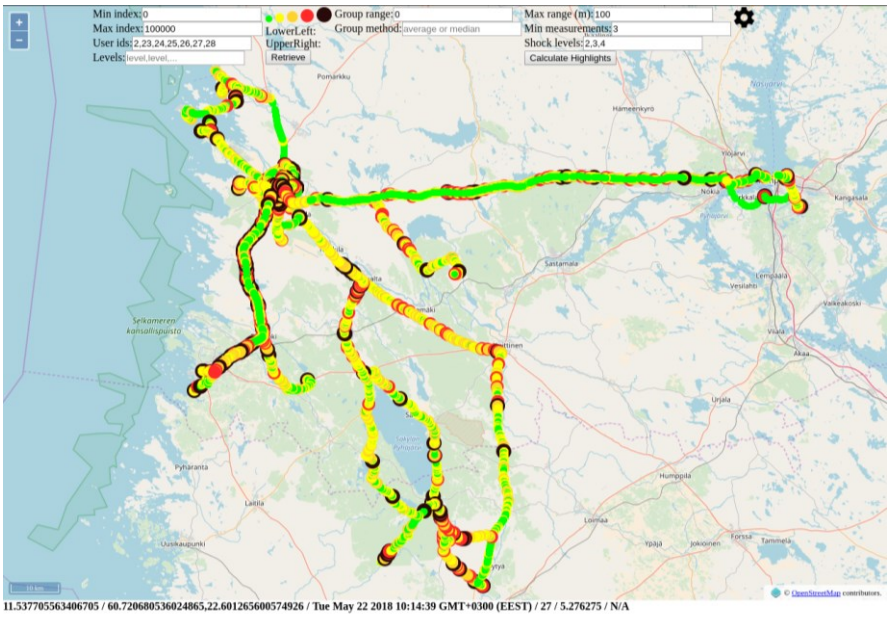


Fig. 6. Visualization of routes driven.

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)